Universidad del Estado de Sonora

División de Ciencias Exactas y Naturales.

Licenciatura en Física.

Física Computacional 1

Hedwin Aaron Encinas Acosta

1. Introduccion

Para esta actividad retomaremos el tema del péndulo, visto en actividades anteriores. El péndulo en el mundo de la física es uno de los sistemas mas interesantes que se puede encontrar, en esta actividad mostraremos el movimiento del péndulo por medio de una animación echa en el lenguaje de programación Python. Se utilizara el péndulo simple como ejemplo ya que es el mas sencillo de representar.

1.1. Péndulos simple

El péndulo simple es una idealización del péndulo normal pero en un sistema aislado y se hacen las siguientes asunciones:

- La masa de la cuerda de la que la masa cuelga es despreciable;
- La lenteja es una masa puntual;
- El movimiento solo ocurre en dos dimensiones;
- El movimiento no pierde energía por fricción o resistencia del aire;
- El campo gravitacional es uniforme;
- El soporte no se mueve;

La ecuación diferencial que representa el movimiento de un péndulo simple es:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0 \tag{1}$$

Donde g es la aceleración debido a la gravedad, l es la longitud del péndulo y θ es el angulo de desplazamiento.[2]

2. Actividad

Para realizar esta actividad utilizaremos un código de ejemplo que se encuentra en la pagina *Pythonic Perambulations*[2]. Este código nos muestra como animar un péndulo doble, utilizando este código como base para animar el péndulo simple. También animaremos el respectivo espacio fase del estado del péndulo.

3. Código

3.1. Código de animación del péndulo

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

```
from scipy.integrate import odeint
import matplotlib.animation as animation
from matplotlib.lines import Line2D

class DoublePendulum:

    def __init__(self,
                 init_state = [120, 0, -20, 0],
                 L1=1.0,  # longitud del pendilo 1
                 L2=0,
                 M1=1.0,  # masa del pendulo 1
                 M2=0,  #
                 G=9.8,  #
                 origin=(0, 0)):
        self.init_state = np.asarray(init_state, dtype='float')
        self.params = (L1, L2, M1, M2, G)
        self.origin = origin
        self.time_elapsed = 0

        self.state = self.init_state * np.pi / 180.

    def position(self):
        (L1, L2, M1, M2, G) = self.params

        x = np.cumsum([self.origin[0],
                       L1 * sin(self.state[0]),
                       L2 * sin(self.state[2])])
        y = np.cumsum([self.origin[1],
                       -L1 * cos(self.state[0]),
                       -L2 * cos(self.state[2])])
        return (x, y)

    def energy(self):
        (L1, L2, M1, M2, G) = self.params

        x = np.cumsum([L1 * sin(self.state[0]),
                       L2 * sin(self.state[2])])
        y = np.cumsum([-L1 * cos(self.state[0]),
                       -L2 * cos(self.state[2])])
        vx = np.cumsum([L1 * self.state[1] * cos(self.state[0]),
                        L2 * self.state[3] * cos(self.state[2])])
        vy = np.cumsum([L1 * self.state[1] * sin(self.state[0]),
                        L2 * self.state[3] * sin(self.state[2])])

        U = G * (M1 * y[0] + M2 * y[1])
        K = 0.5 * (M1 * np.dot(vx, vx) + M2 * np.dot(vy, vy))
```

```
        return U + K

    def dstate_dt(self, state, t):
        (M1, M2, L1, L2, G) = self.params

        dydx = np.zeros_like(state)
        dydx[0] = state[1]
        dydx[2] = state[3]

        cos_delta = cos(state[2] - state[0])
        sin_delta = sin(state[2] - state[0])

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


        return dydx

    def step(self, dt):
        self.state = integrate.odeint(self.dstate_dt, self.state, [0, dt])[1]
        self.time_elapsed += dt




#-----------------------------------------------------------
#Damos las condiciones iniciales
pendulum = DoublePendulum([90., 0, 0.0, 0.0])
dt = 1./30 # 30 fps

#-----------------------------------------------------------
# Figura y animacion


fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal', autoscale_on=False,
                     xlim=(-2, 2), ylim=(-2, 2))

ax.grid()
line, = ax.plot([], [], 'o-', lw=2)
time_text = ax.text(0.02, 0.95, '', transform=ax.transAxes)
energy_text = ax.text(0.02, 0.90, '', transform=ax.transAxes)
```

```
def init():
    line.set_data([], [])
    time_text.set_text('')
    energy_text.set_text('')
    return line,time_text, energy_text

def animate(i):
    global pendulum, dt
    pendulum.step(dt)

    line.set_data(*pendulum.position())
    time_text.set_text('time = %.1f' % pendulum.time_elapsed)
    energy_text.set_text('energy = %.3f J' % pendulum.energy())
    return line,time_text, energy_text

# choose the interval based on dt and the time to animate one step
from time import time
t0 = time()
animate(0)
t1 = time()
interval = 1000 * dt - (t1 - t0)

ani = animation.FuncAnimation(fig, animate, frames=300,
                              interval=interval, blit=True, init_func=init,)



plt.show()
```

3.2. Código de animación del espacio fase

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from matplotlib.lines import Line2D
import matplotlib.animation as animation
#Definimos las constantes

g = 9.8
l = 1.0
b = 0.0
c = g/l

#Damos las condiciones iniciales
X1 =np.array([(180/180.0)*np.pi,(90/180.0)*np.pi])
```

```
t = np.linspace(-0.0*np.pi,5.0*np.pi,500)

#Definimos la ecuacion diferencial del pendulo
def p (y, t, b, c):
    theta, omega = y
    dy_dt = [omega,-b*omega -c*np.sin(theta)]
    return dy_dt

#Trayectoria
y0 =  X1
X = odeint(p, y0, t, args=(b,c))


class SubplotAnimation(animation.TimedAnimation):
    def __init__(self):
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)


        self.t = np.linspace(0, 80, 400)
        self.x = X[:,0]
        self.y = X[:,1]

        ax1.set_xlabel('x')
        ax1.set_ylabel('y')
        self.line1 = Line2D([], [], color='blue')
        self.line1a = Line2D([], [], color='red', linewidth=2)
        self.line1e = Line2D(
            [], [], color='red', marker='o', markeredgecolor='r')
        ax1.add_line(self.line1)
        ax1.add_line(self.line1a)
        ax1.add_line(self.line1e)
        ax1.set_xlim(-10, 10)
        ax1.set_ylim(-10, 10)
        ax1.set_aspect('equal', 'datalim')

        animation.TimedAnimation.__init__(self, fig, interval=50, blit=True)

    def _draw_frame(self, framedata):
        i = framedata
        head = i - 1
        head_len = 10
        head_slice = (self.t > self.t[i] - 1.0) & (self.t < self.t[i])

        self.line1.set_data(self.x[:i], self.y[:i])
        self.line1a.set_data(self.x[head_slice], self.y[head_slice])
```

```
        self.line1e.set_data(self.x[head], self.y[head])


        self._drawn_artists = [self.line1, self.line1a, self.line1e]

    def new_frame_seq(self):
        return iter(range(self.t.size))

    def _init_draw(self):
        lines = [self.line1, self.line1a, self.line1e]

        for l in lines:
            l.set_data([], [])

ani = SubplotAnimation()
#ani.save('test_sub.mp4')
plt.show()
```

4. Animaciones para diferentes casos

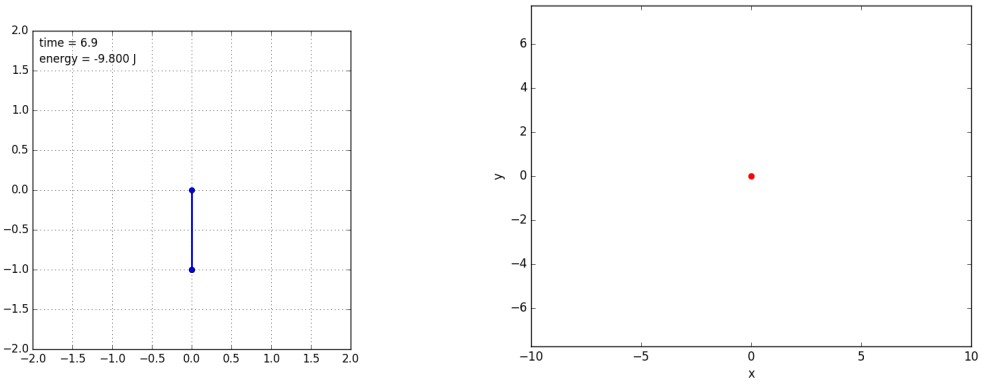

Figura 1: Imagen de Animación de péndulo con ángulo inicial 0 y velocidad angular 0.

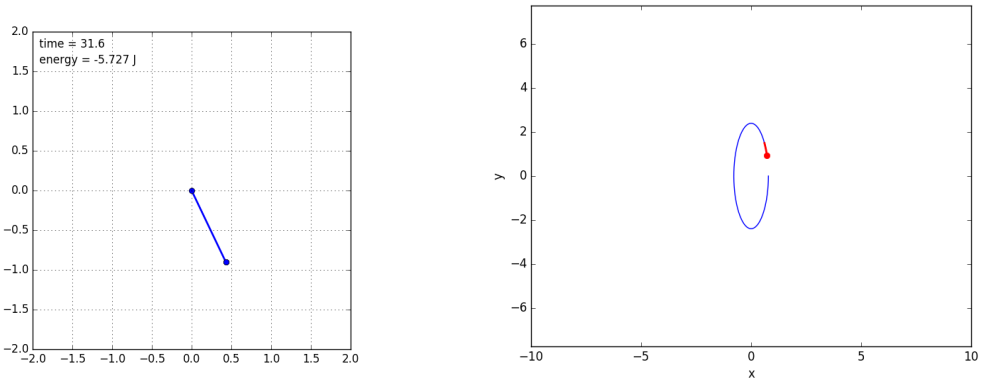

Figura 2: Imagen de Animación de péndulo con ángulo inicial 45 y velocidad angular 0.

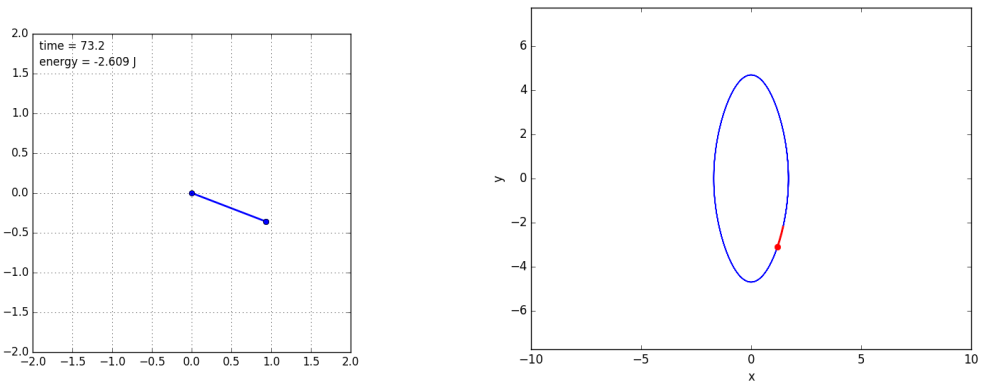

Figura 3: Imagen de Animación de péndulo con ángulo inicial 90 y velocidad angular 0.

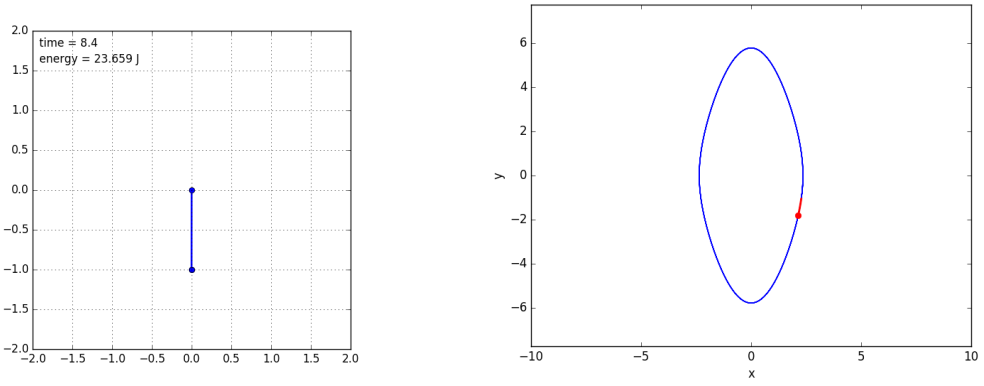

Figura 4: Imagen de Animación de péndulo con ángulo inicial 135 y velocidad angular 0.

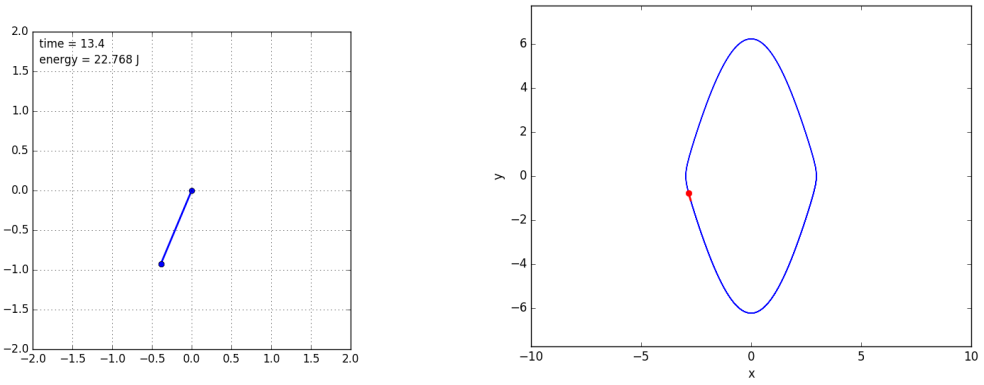

Figura 5: Imagen de Animación de péndulo con ángulo inicial 170 y velocidad angular 0.

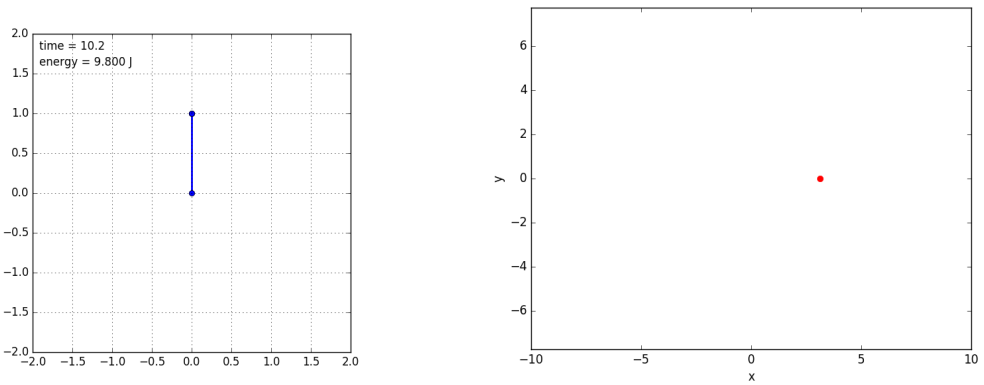

Figura 6: Imagen de Animación de péndulo con ángulo inicial 180 y velocidad angular 0.

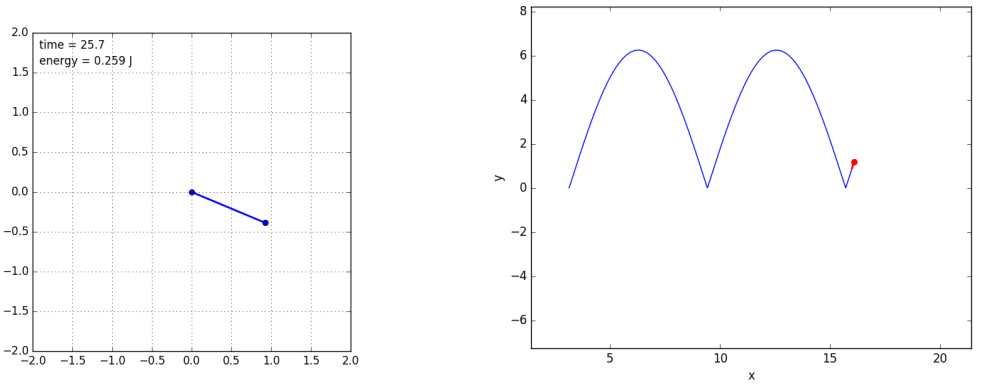

Figura 7: Péndulo con apenas suficiente energía para una vuelta

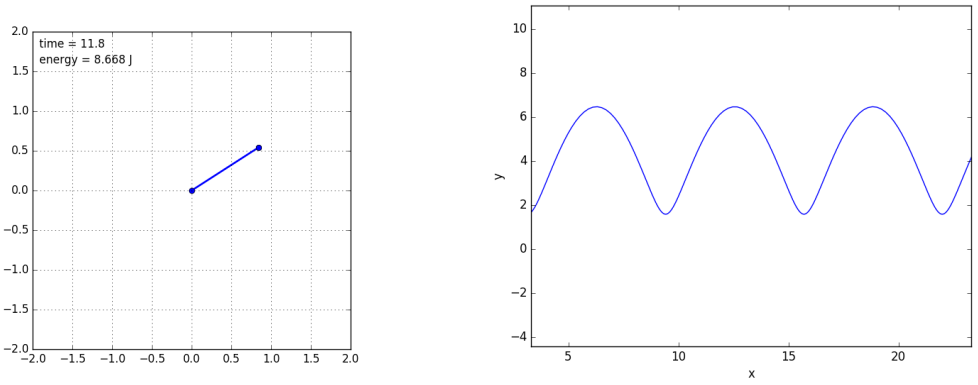

Figura 8: Péndulo con suficiente energía para una vuelta

Referencias

[1] Wikipedia, Pendulum
https://en.wikipedia.org

[2] Pythonic Perambulations
https://jakevdp.github.io/