Universidad de Sonora
División de Ciencias Exactas y Naturales
Licenciatura en Física
Física computacional
2016-1

Actividad 10: Animaciones con Matplotlib

Carolina Valenzuela Córdova

1 de Mayo de 2016

Índice

1. Descripción de la actividad

Para esta actividad, se nos solicitó modificar un código ejemplo proporcionado por el profesor, de tal manera que pudiéramos graficar ciertas animaciones propuestas por un artículo del péndulo en Wikipedia.
Cabe mencionar que dicho código estaba diseñado para animar un péndulo doble, y el artículo sugerido contaba con animaciones de péndulo simple. Esta fue la principal modificación que se tuvo que realizar; además tuvimos que presentar 8 casos de diferentes ángulos iniciales.
Obtuvimos 8 animaciones de péndulo en estos casos y su diagrama de espacio fase correspondiente. Únicamente modificamos la parte del código donde se encuentran estas condiciones.
el código utilizado fue el siguiente:

```
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([(180.0/180.0)*np.pi,(-250./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()
```

Vemos que las modificaciones hechas fueron para lograr que el péndulo fuera simple, sin fricción y se pudieran graficar tanto las animaciones correspondientes a los casos propuestos por Wikipeia, como los diagramas de espacio fase.

2. Resultados

Aquí se presentan screenshots de las animaciones obtenidas en la actividad, tanto del péndulo como de su espacio fase. Para cada uno de los casos, se modificaron sus condiciones iniciales establecidas por el código, especificando el ángulo en cada uno de ellos.

2.1. Caso 1: $\theta_0 = 0^o$

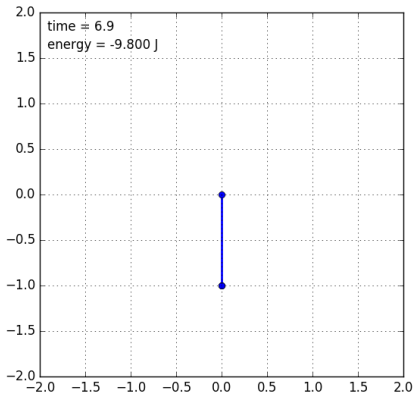


Figura 1: Para$\theta_0 = 0$

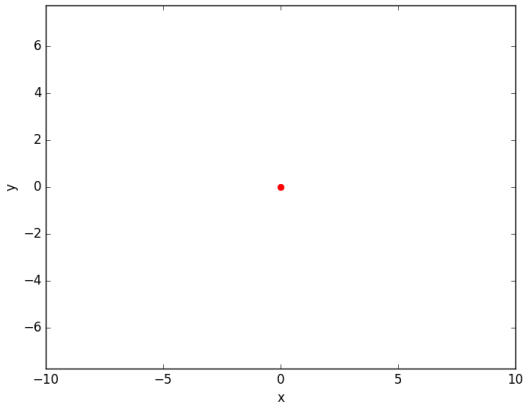


Figura 2: Para$\theta_0 = 0$

2.2. Caso 2: $\theta_0 = 45^o$

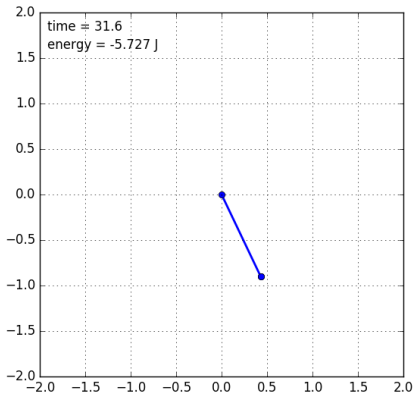


Figura 3: Para$\theta_0 = 45^o$

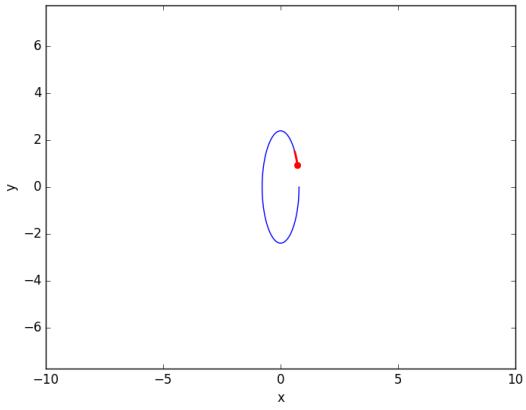


Figura 4: Para$\theta_0 = 45^o$

2.3. Caso3: $\theta_0 = 90^0$

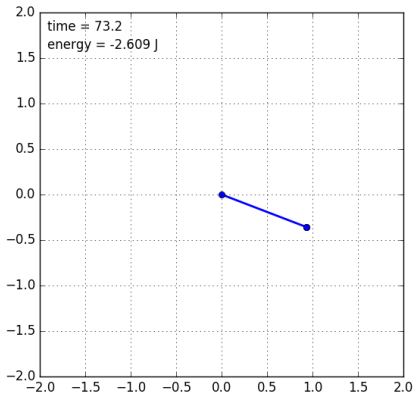


Figura 5: Para$\theta_0 = 90^o$

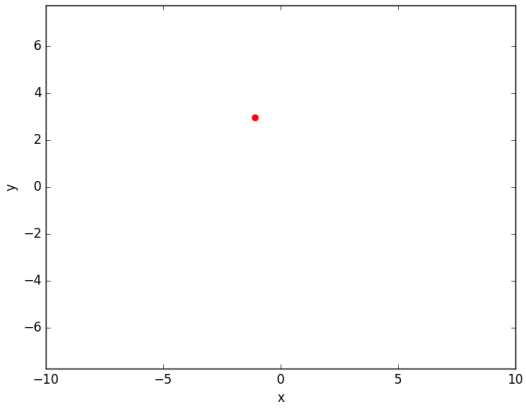


Figura 6: Para$\theta_0 = 90^o$

2.4. Caso4: $\theta_0 = 135^0$

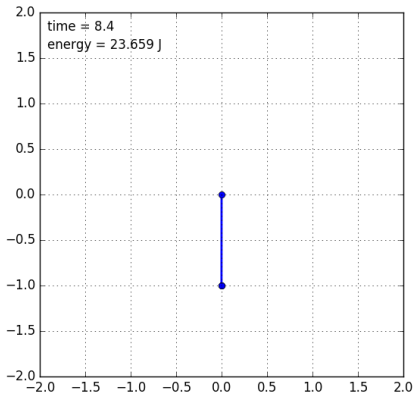


Figura 7: Para$\theta_0 = 135^o$

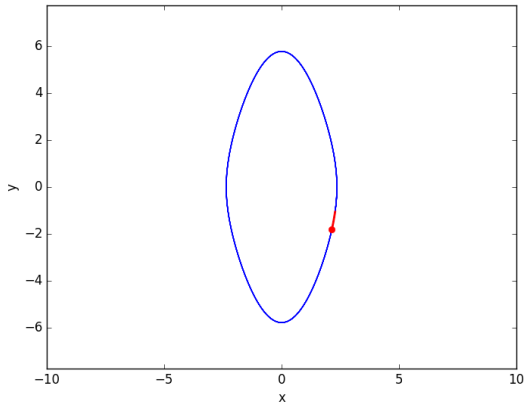


Figura 8: Para$\theta_0 = 135^o$

2.5. Caso 5: $\theta_0 = 170^0$

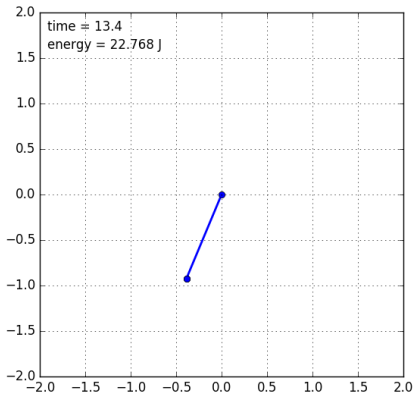


Figura 9: Para$\theta_0 = 170^o$

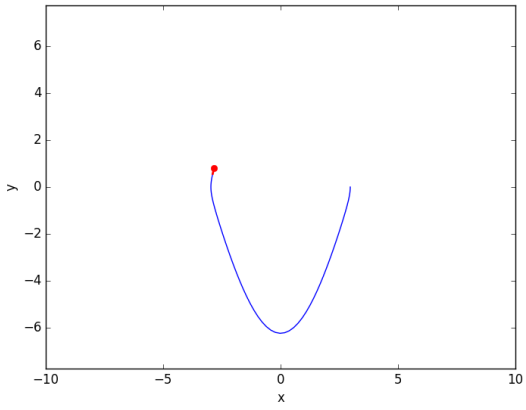


Figura 10: Para$\theta_0 = 170^o$

2.6. Caso 6: $\theta_0 = 180^0$

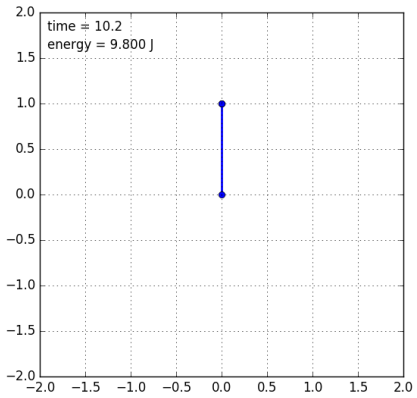


Figura 11: Para$\theta_0 = 180^o$

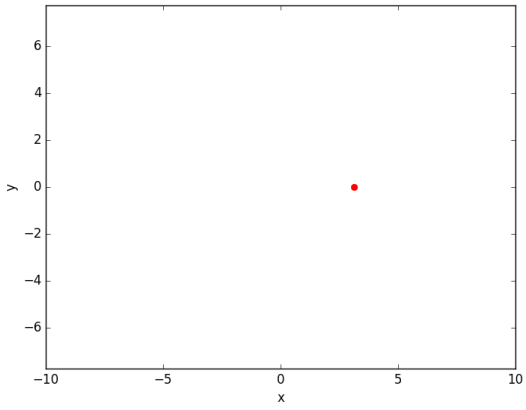
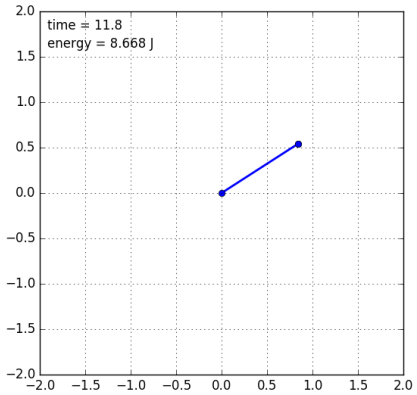
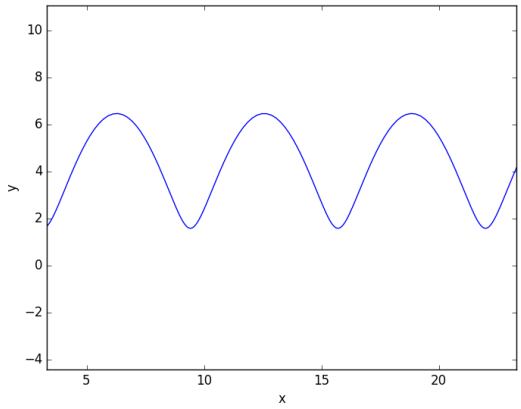


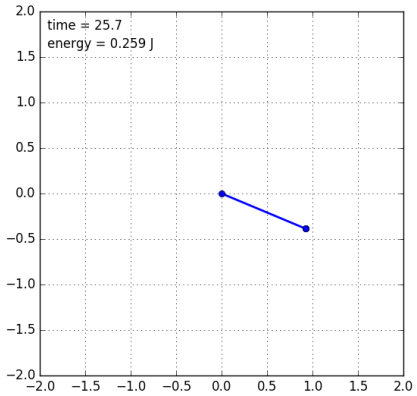
Figura 12: Para$\theta_0 = 180^o$

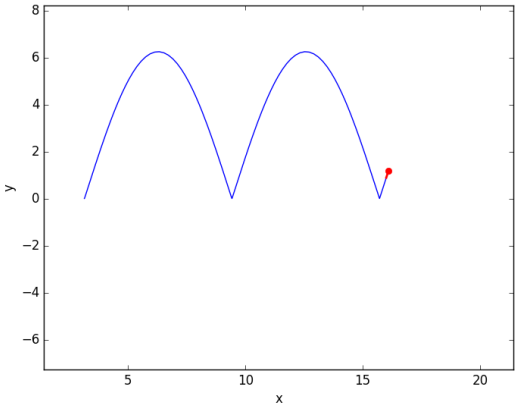
2.7. Caso 7: Péndulo con energía apenas suficiente para dar una vuelta completa





2.8. Caso 8: Péndulo con suficiente energía para dar una vuelta completa





3. Conclusión

Para esta actividad fue bastante lo que se tuvo que observar y manipular para entender cómo se encontraban distribuidas las variables y cómo es que este código opera.
Pudimos aprender a producir nuestras propias animaciones para fenómenos físicos, siendo necesaria la comprensión mecánica de lo que se estaba simulando para estar conciente de si el código funcionaba correctamente. Los archivos

de video que muestran los resultados obtenidos de nuestras animaciones se encuentran en la carpeta de repositorio de Github, junto a este reporte.