

Actividad 10: Animaciones con Matplotlib

Martin Alejandro Paredes Sosa

Marzo, 2016

1. Introducción

La matemática de un péndulo simple es, en general, compleja. Hacer suposiciones que simplifican la descripción nos permite resolver analíticamente las ecuaciones de movimiento. El péndulo simple, es una idealización de un péndulo real, pero en un sistema aislado donde se asume:

- La cuerda tiene una masa despreciable, es rígida y se mantiene tensa.
- El péndulo se maneja como una masa puntual.
- El movimiento es en dos dimensiones trazando un arco.
- No pierde energía por fricción o resistencia al aire.
- El campo gravitacional es uniforme.
- El soporte no se mueve.

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

$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell} \sin \theta = 0 \quad (1)$$

donde g es la aceleración de la gravedad, ℓ es la longitud del péndulo y θ es el ángulo de desplazamiento [1].

En esta práctica se nos pidió realizar una animación de un péndulo utilizando la biblioteca de `Matplotlib`. Haciendo uso del código de Jake Vanderplas para el péndulo doble [2] y el código de Matplotlib de subplots[3], se crearon las animaciones del péndulo y su espacio fase.

2. Ejercicio y Resultados

Esta actividad consistió en realizar un código en python que nos permitiera construir una animación del péndulo simple. Se hizo uso de la librería `matplotlib.animation` para crear la animación y poder observar el movimiento del péndulo con diferentes condiciones iniciales.

Los códigos que se utilizaron fueron los siguientes:

```

1 import numpy as np
2 from matplotlib import pyplot as plt
3 from matplotlib import animation as an
4 from matplotlib.lines import Line2D
5 from scipy.integrate import odeint
6
7 #Definiendo las constantes de la Ec. Diferencial
8 g = 9.81
9 l = 1.0
10 b = 0.0 #Pendulo simple por lo tanto sin friccion
11 c = g/l
12
13 #Codiciones Iniciales
14 X_f1 =np.array([(170.0/180.0)*np.pi,(0./180.0)*np.pi])
15 t = np.linspace(-0.0*np.pi,5.0*np.pi,500) #Para generar la solucion
16
17 #Definicion de la ecuacion diderencial del pendulo
18 def p (y, t, b, c):
19     theta, omega = y
20     dy_dt = [omega,-b*omega -c*np.sin(theta)]
21     return dy_dt
22
23 #=====
24 # Trayectoria
25 y0 = X_f1 # Punto de Inicio
26 X = odeint(p, y0, t, args=(b,c))
27
28 #=====
29 #Graficar
30
31 class SubplotAnimation(an.TimedAnimation):
32     def __init__(self):
33         fig = plt.figure()
34         ax1 = fig.add_subplot(1, 1, 1)
35
36         self.t = np.linspace(0, 80, 400)
37         self.x = X[:,0]
38         self.y = X[:,1]
39
40         self.line1 = Line2D([], [], color='black')
41         self.line1a = Line2D([], [], color='red', linewidth=2)
42         self.line1e = Line2D(
43             [], [], color='red', marker='o', markeredgecolor='r')
44         ax1.add_line(self.line1)
45         ax1.add_line(self.line1a)
46         ax1.add_line(self.line1e)
47         ax1.set_xlim(-10, 10)
48         ax1.set_ylim(-10, 10)
49         ax1.set_aspect('equal', 'datalim')

```

```

50         an.TimedAnimation.__init__(self, fig, interval=50, blit=True)
51
52
53     def _draw_frame(self, framedata):
54         i = framedata
55         head = i - 1
56         #head_len = 10
57         head_slice = (self.t > self.t[i] - 1.0) & (self.t < self.t[i])
58
59         self.line1.set_data(self.x[:i], self.y[:i])
60         self.line1a.set_data(self.x[head_slice], self.y[head_slice])
61         self.line1e.set_data(self.x[head], self.y[head])
62
63     def new_frame_seq(self):
64         return iter(range(self.t.size))
65
66     def _init_draw(self):
67         lines = [self.line1, self.line1a, self.line1e]
68         for l in lines:
69             l.set_data([], [])
70
71 ani = SubplotAnimation()
72 #ani.save('test_sub.mp4')
73 plt.show()

```

Listing 1: Código Animacion.Fase.py

```

1 from numpy import sin, cos
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import scipy.integrate as integrate
5 import matplotlib.animation as animation
6
7 class DoublePendulum:
8     """Double Pendulum Class
9
10     init_state is [theta1, omega1, theta2, omega2] in degrees,
11     where theta1, omega1 is the angular position and velocity of the first
12     pendulum arm, and theta2, omega2 is that of the second pendulum arm
13     """
14     def __init__(self,
15                 init_state = [120, 0, 0, 0],
16                 L1=1.0, # length of pendulum 1 in m
17                 L2=0.0, # length of pendulum 2 in m
18                 M1=1.0, # mass of pendulum 1 in kg
19                 M2=1.0, # mass of pendulum 2 in kg
20                 G=9.8, # acceleration due to gravity, in m/s^2
21                 origin=(0, 0)):
22         self.init_state = np.asarray(init_state, dtype='float')
23         self.params = (L1, L2, M1, M2, G)

```

```

24     self.origin = origin
25     self.time_elapsed = 0
26
27     self.state = self.init_state * np.pi / 180.
28
29     def position(self):
30         """compute the current x,y positions of the pendulum arms"""
31         (L1, L2, M1, M2, G) = self.params
32
33         x = np.cumsum([self.origin[0],
34                        L1 * sin(self.state[0]),
35                        L2 * sin(self.state[2])])
36         y = np.cumsum([self.origin[1],
37                        -L1 * cos(self.state[0]),
38                        -L2 * cos(self.state[2])])
39         return (x, y)
40
41     def energy(self):
42         """compute the energy of the current state"""
43         (L1, L2, M1, M2, G) = self.params
44
45         x = np.cumsum([L1 * sin(self.state[0]),
46                        L2 * sin(self.state[2])])
47         y = np.cumsum([-L1 * cos(self.state[0]),
48                        -L2 * cos(self.state[2])])
49         vx = np.cumsum([L1 * self.state[1] * cos(self.state[0]),
50                         L2 * self.state[3] * cos(self.state[2])])
51         vy = np.cumsum([L1 * self.state[1] * sin(self.state[0]),
52                         L2 * self.state[3] * sin(self.state[2])])
53
54         U = G * (M1 * y[0] + M2 * y[1])
55         K = 0.5 * (M1 * np.dot(vx, vx) + M2 * np.dot(vy, vy))
56
57         return U + K
58
59     def dstate_dt(self, state, t):
60         """compute the derivative of the given state"""
61         (M1, M2, L1, L2, G) = self.params
62
63         dydx = np.zeros_like(state)
64         dydx[0] = state[1]
65         dydx[2] = state[3]
66
67         cos_delta = cos(state[2] - state[0])
68         sin_delta = sin(state[2] - state[0])
69
70         den1 = (M1 + M2) * L1 - M2 * L1 * cos_delta * cos_delta
71         dydx[1] = (M2 * L1 * state[1] * state[1] * sin_delta * cos_delta
72                  + M2 * G * sin(state[2]) * cos_delta

```

```

73         + M2 * L2 * state[3] * state[3] * sin_delta
74         - (M1 + M2) * G * sin(state[0])) / den1
75
76     den2 = (L2 / L1) * den1
77     dydx[3] = (-M2 * L2 * state[3] * state[3] * sin_delta * cos_delta
78               + (M1 + M2) * G * sin(state[0]) * cos_delta
79               - (M1 + M2) * L1 * state[1] * state[1] * sin_delta
80               - (M1 + M2) * G * sin(state[2])) / den2
81
82     return dydx
83
84     def step(self, dt):
85         """execute one time step of length dt and update state"""
86         self.state = integrate.odeint(self.dstate_dt, self.state, [0, dt])
87         self.time_elapsed += dt
88
89     #-----
90     # set up initial state and global variables
91     pendulum = DoublePendulum([180.0, -250.0, 0.0, 0.0])
92     dt = 1./60. # 60 fps
93
94     #-----
95     # set up figure and animation
96     fig = plt.figure()
97     ax = fig.add_subplot(111, aspect='equal', autoscale_on=False,
98                         xlim=(-2, 2), ylim=(-2, 2))
99     ax.grid()
100
101     line, = ax.plot([], [], 'o-', lw=2)
102     time_text = ax.text(0.02, 0.95, '', transform=ax.transAxes)
103     energy_text = ax.text(0.02, 0.90, '', transform=ax.transAxes)
104
105     def init():
106         #initialize animation
107         line.set_data([], [])
108         time_text.set_text('')
109         energy_text.set_text('')
110         return line, time_text, energy_text
111
112     def animate(i):
113         #perform animation step
114         global pendulum, dt
115         pendulum.step(dt)
116
117         line.set_data(*pendulum.position())
118         time_text.set_text('time = %.1f' % pendulum.time_elapsed)
119         #energy_text.set_text('energy = %.3f J' % pendulum.energy())
120         return line, time_text, energy_text

```

```

121
122 # choose the interval based on dt and the time to animate one step
123 from time import time
124 t0 = time()
125 animate(0)
126 t1 = time()
127 interval = 1000 * dt - (t1 - t0)
128
129 ani = animation.FuncAnimation(fig, animate, frames=300,
130                             interval=interval, blit=True, init_func=init
131                             )
132 plt.show()

```

Listing 2: Código Animacion_Pendulo.py

El resultado que se obtuvo fue el siguiente:

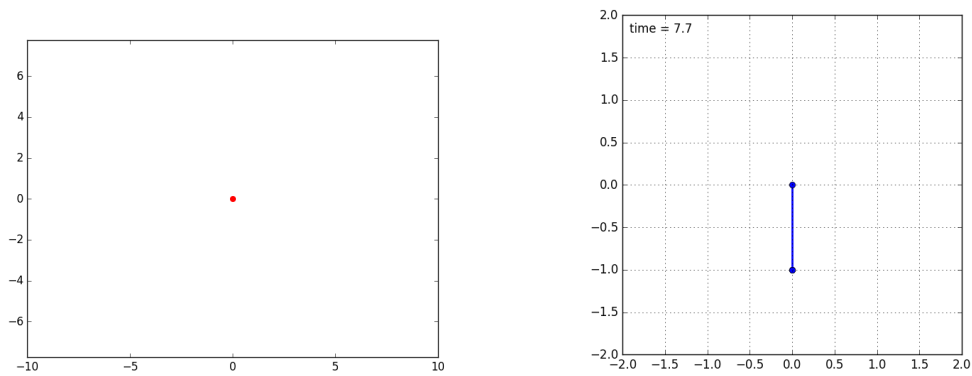


Figura 1: Imagen de Animación de péndulo con ángulo inicial 0.

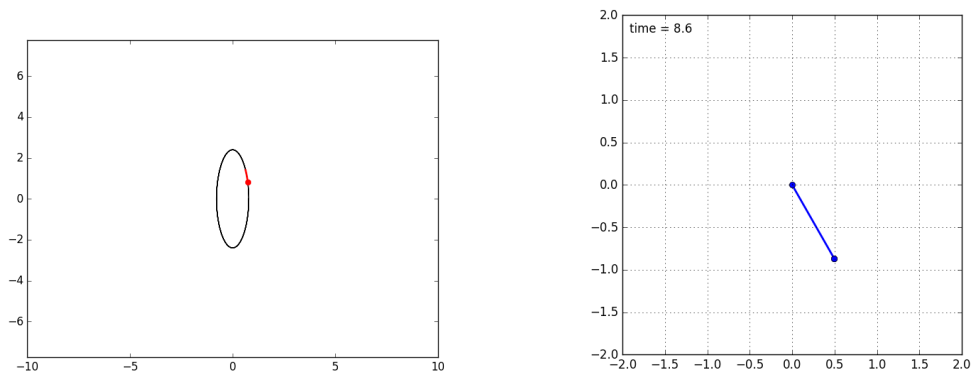


Figura 2: Imagen de Animación de péndulo con ángulo inicial 45.

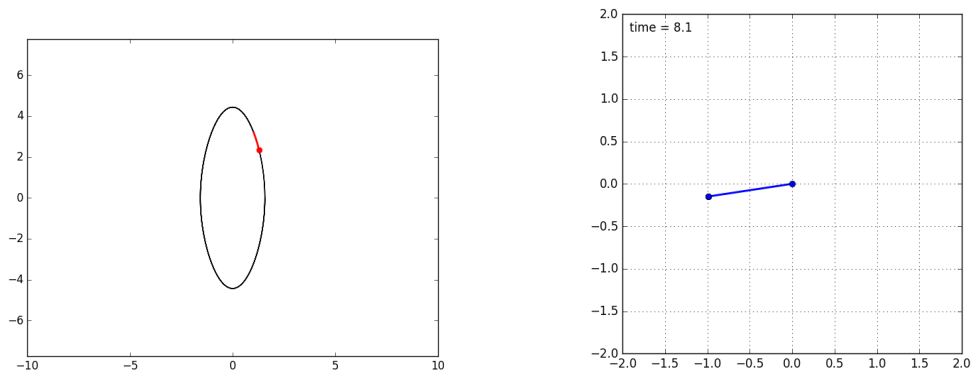


Figura 3: Imagen de Animación de péndulo con ángulo inicial 90.

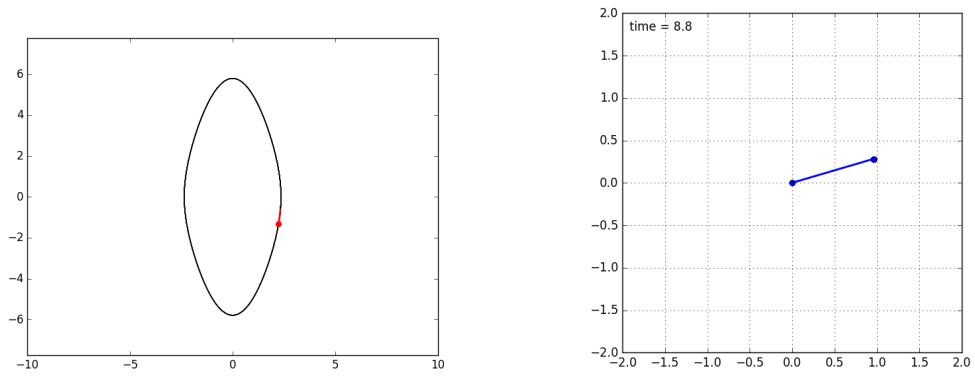


Figura 4: Imagen de Animación de péndulo con ángulo inicial 135.

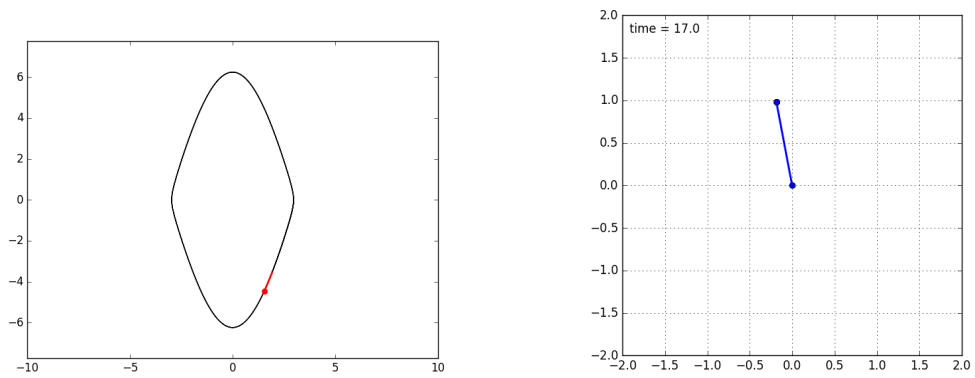


Figura 5: Imagen de Animación de péndulo con ángulo inicial 170.

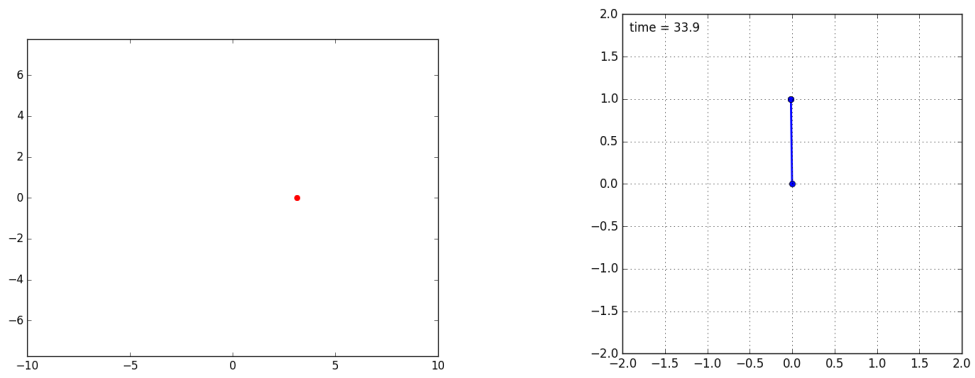


Figura 6: Imagen de Animación de péndulo con ángulo inicial 180.

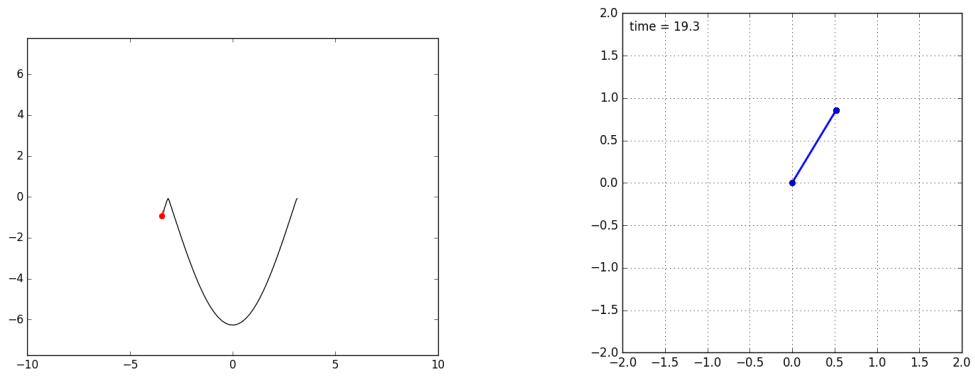


Figura 7: Imagen de Animación de péndulo con apenas energía para completar la vuelta.

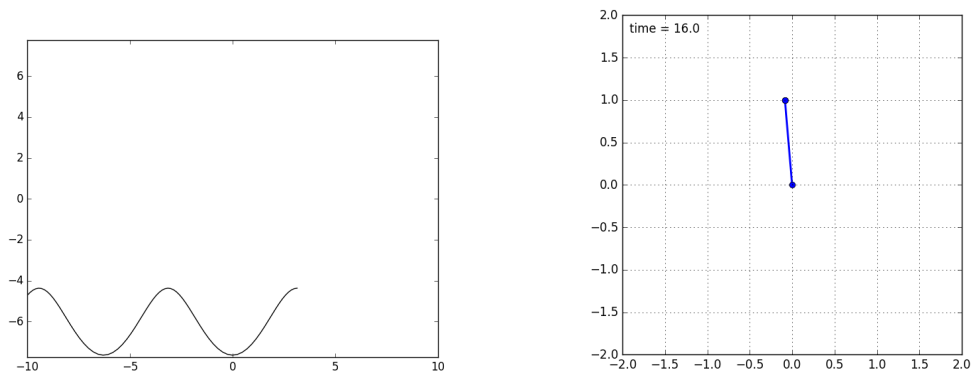


Figura 8: Imagen de Animación de péndulo con suficiente energía para completar la vuelta.

Se adjuntaron vídeos de las animaciones de cada uno de los casos.

Referencias

- [1] Wikipedia,(2016) *Pendulum (mathematics)*. Recuperado de https://en.wikipedia.org/wiki/Pendulum_%28mathematics%29
- [2] Vanderplas, J.(2012) *Matplotlib Animation Tutorial* Recuperado de <https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/>
- [3] Matplotlib(2012) *animation example code: subplots.py* Recuperado de <http://matplotlib.org/examples/animation/subplots.html>
- [4] Lizárraga, C. (2016) *Actividad 10 (2016-1)*. Recuperado de [http://computacional1.pbworks.com/w/page/107247876/Actividad%2010%20\(2016-1\)](http://computacional1.pbworks.com/w/page/107247876/Actividad%2010%20(2016-1))