Actividad 10: Animaciones con Matplotlib.

Valenzuela Carrillo María Inés

27 de Abril del 2016

1. Introducción

Las animaciones permiten visualizar cosas de una forma más realista y más dinámica, el objetivo de la actividad es aprender a reproducir animaciones apoyados en la biblioteca de Matplotlib de Python.

Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar a la de MATLAB.[1]

En varias actividades se ha utilizado el Artículo del Péndulo de la Wikipedia. En este caso se pide reproducir animaciones del movimiento en el espacio fase y el movimiento en el espacio físico del péndulo, similares a los 8 casos que aparecen en la Wikipedia.

En internet se pueden encontrar códigos para realizar animaciones, como los siguientes ejemplos:

- Matplotlib animation examples: double_pendulum_animated.py
- Matplotlib Animation Tutorial.

Basándose en los ejemplos de las animaciones con Matplotlib, se pide adaptar cualquiera de los códigos.

2. Animaciones Del Péndulo

A continuación se presenta el código realizado para obtener las animaciones para diversos ángulos del péndulo.

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

#Movimiento Fisico del Pendulo

class DoublePendulum:

    def __init__(self,
                 init_state = [ 120 ,0, 0, 0],
                 L1=1.0,     # Largo del pendulo en
                    metros
                 L2=0.0,     # Ponemos 0 el largo del
                    segundo pendulo
                 M1=1.0,     # Masa del primer pendulo
                 M2=1.0,     # Masa del segundo pendulo
                 G=9.8,      # Aceleracion gravitacional
                 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])])
        y = np.cumsum([self.origin[1],
```

```
33                          -L1 * cos(self.state[0])])
34          return (x, y)
35
36
37      def dstate_dt(self, state, t):
38
39          (M1, M2, L1, L2, G) = self.params
40
41          dydx = np.zeros_like(state)
42          dydx[0] = state[1]
43          dydx[2] = state[3]
44
45          cos_delta = cos(state[2] - state[0])
46          sin_delta = sin(state[2] - state[0])
47
48          den1 = (M1 + M2) * L1 - M2 * L1 * cos_delta *
               cos_delta
49          dydx[1] = (M2 * L1 * state[1] * state[1] *
               sin_delta * cos_delta
50                  + M2 * G * sin(state[2]) * cos_delta
51                  + M2 * L2 * state[3] * state[3] *
                       sin_delta
52                  - (M1 + M2) * G * sin(state[0])) /
                       den1
53
54          return dydx
55
56
57      def step(self, dt):
58          """execute one time step of length dt and update
                state"""
59          self.state = integrate.odeint(self.dstate_dt,
               self.state, [0, dt])[1]
60          self.time_elapsed += dt
61
62          return self.state
63
64
65  # Estado inicial
66  pendulum = DoublePendulum([170, 40, 0., 0.0])   #theta1,
```

```
    omega1, theta2, omega2
dt = 1./30


# Figuara 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, color='magenta')
time_text = ax.text(0.02, 0.95, '', transform=ax.
   transAxes)



def init():
    """initialize animation"""
    line.set_data([], [])
    time_text.set_text('')
    return line, time_text


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

    line.set_data(*pendulum.position())
    time_text.set_text('tiempo = %.1f' % pendulum.
       time_elapsed)
    return line, time_text


from time import time
t0 = time()
animate(0)
t1 = time()
```

```
103 interval = 1000 * dt - (t1 - t0)
104
105 ani = animation.FuncAnimation(fig, animate, frames=80,
106                                interval=interval, blit=
                                      True, init_func=init)
107
108 ani.save('pendulo0', writer='ffmpeg')
109
110
111 plt.show()
112
113 #=======================================================
114
115 # ESPACIO FASE
116
117 def pend(y, t, b, c):
118      theta, omega = y
119      dydt = [omega, -b*omega - c*np.sin(theta)]
120      return dydt
121
122 #Parametros
123 b = 0.0          #Friccion
124 g = 9.8          #Aceleracion gravitacional
125 l = 4.5          #longitud de la cuerda
126 c = g/l
127
128
129 y0 = [(0*np.pi)/180, 15.0]      #Angulo inical,
       velocidad angular
130
131 t = np.linspace(0, 20, 100)
132
133 from scipy.integrate import odeint
134 sol = odeint(pend, y0, t, args=(b, c))
135
136
137 #Para crear los archivos:
138
139 file = open("90.txt","w")
140 file.close()
```

```
141 file = open('90.txt','r')
142
143 #Para crear archivo a partir de los datos obtenidos
144 np.savetxt("90.txt", sol)
145
146 file = open("90.txt","r")
147 print file.read()
148
149 import matplotlib.pyplot as plt
150 import numpy as np
151
152
153 data = np.loadtxt('90.txt')
154
155 x1=data[:,0]    #velocidad angular
156 y1=data[:,1]    #posicion angular
157
158
159 x11 = x1.astype(np.float)       #velocidad
160 y11 = y1.astype(np.float)      #posicion
161
162
163 #Animacion del espacio fase
164
165 import numpy as np
166 import matplotlib.pyplot as plt
167 from matplotlib.lines import Line2D
168 import matplotlib.animation as animation
169
170
171
172 class SubplotAnimation(animation.TimedAnimation):
173     def __init__(self):
174         fig = plt.figure()
175
176         ax1 = fig.add_subplot(1, 1, 1)
177
178
179
180         #Funcion a graficar
```

```
181         self.t = np.linspace(0, 80, 300)#tiempo inicial
               ,velocidad inicial, puntos
182         self.x = x11                #funcion eje x
183         self.y = y11                #funcin eje y
184         self.z = 5 * self.t
185 
186 
187         #Caracteristicas animacion eje xy
188 
189         ax1.set_xlabel('Posicion angular')
190         ax1.set_ylabel('Velocidad angular')
191         self.line1 = Line2D([], [], color='pink')
192         self.line1a = Line2D([], [], color='magenta',
               linewidth=2)
193         self.line1e = Line2D(
194             [], [], color='magenta', marker='o',
                   markeredgecolor='b')
195         ax1.add_line(self.line1)
196         ax1.add_line(self.line1a)
197         ax1.add_line(self.line1e)
198         ax1.set_xlim(-4, 4)
199         ax1.set_ylim(-8, 8)
200 
201 
202 
203         animation.TimedAnimation.__init__(self, fig,
               interval=115, blit=115)
204 
205     def _draw_frame(self, framedata):
206         i = framedata
207         head = i - 1
208         head_len = 10
209         head_slice = (self.t > self.t[i] - 1.0) & (self.
               t < self.t[i])
210 
211         self.line1.set_data(self.x[:i], self.y[:i])
212         self.line1a.set_data(self.x[head_slice], self.y[
               head_slice])
213         self.line1e.set_data(self.x[head], self.y[head])
214 
```

```
215         self._drawn_artists = [self.line1, self.line1a,
               self.line1e,
216                                #self.line2, self.line2a,
                                       self.line2e,
217                                #self.line3, self.line3a,
                                       self.line3e
218                                ]
219
220     def new_frame_seq(self):
221         return iter(range(self.t.size))
222
223     def _init_draw(self):
224         lines = [self.line1, self.line1a, self.line1e,
225                  #self.line2, self.line2a, self.line2e,
226                  #self.line3, self.line3a, self.line3e
227                  ]
228         for l in lines:
229             l.set_data([], [])
230
231 ani = SubplotAnimation()
232 #ani.save('pendulo0F', writer='ffmpeg')
233
234
235 plt.show()
```

Referencias

[1] Wikipedia, the free encyclopedia, *Matplotlib*, recuperado de https://es.wikipedia.org/wiki/Matplotlib

[2] Actividad 10 (2016-1), recuperado de http://computacional1.pbworks.com/w/page/107247876/Actividad10(2016-1).

[3] Matplotlib, *Matplotlib animation examples: double_pendulum_animated.py*, recuperado de http://matplotlib.org/examples/animation/double_pendulum_animated.html

[4] Pythonic Perambulations, *Matplotlib Animation Tutorial*, recuperado de https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/