Universidad de Sonora

División de Ciencias Exactas y Naturales

Física Computacional

Teoría del Caos

Delgado Curiel Esteban

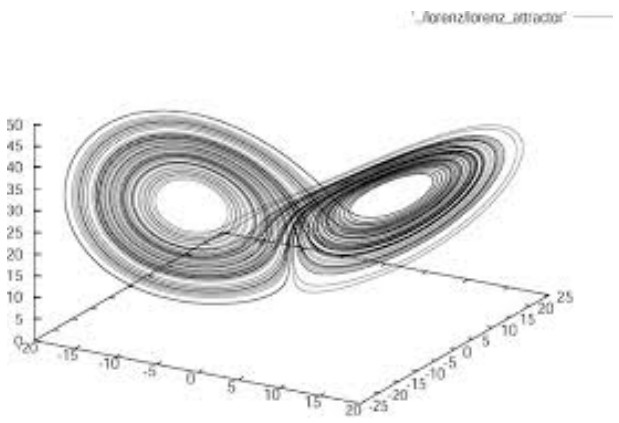
Profesor: Carlos Lizárraga Celaya

13 de Mayo del 2017

Teoría del Caos

Edward Norton Lorenz (Mayo 23, 1917 - Abril 16, 2008) fue un matemático, meteorólogo americano, pionero en la teoría del caos junto a Mary Cartwright. Lorenz introdujo la noción de atractores extraños y creó el término Efecto Mariposa. El profesor Lorenz fue el primero en reconocer lo que se denomina comportamiento caótico en el modelado matemático de los sistemas meteorológicos. A principios de la década de 1960, Lorenz se dio cuenta de que las pequeñas diferencias en un sistema dinámico, como la atmósfera - o un modelo de la atmósfera - podrían desencadenar enormes y, a menudo, insospechados resultados. En meteorología, llegó a la conclusión de que puede ser fundamentalmente imposible hacer predicciones más allá de dos o tres semanas con un grado razonable de exactitud.

El comienzo de la reciente historia del caos se sitúa en la década de 1950 cuando se inventaron los ordenadores y se desarrollaron algunas intuiciones sobre el comportamiento de los sistemas no lineales. Esto es, cuando se vieron las primeras gráficas sobre el comportamiento de estos sistemas mediante métodos numéricos. En 1963 Edward Lorenz trabajaba en unas ecuaciones, las mundialmente conocidas como ecuaciones de Lorenz, que esperaba predijeran el tiempo en la atmósfera, y trató mediante los ordenadores de ver gráficamente el comportamiento de sus ecuaciones. Los ordenadores de aquella época eran muy lentos, por eso se dice que Lorentz fue a tomar un té mientras el ordenador hacía los cálculos, y cuando volvió se encontró con una figura que ahora se conoce como atractor de Lorenz.

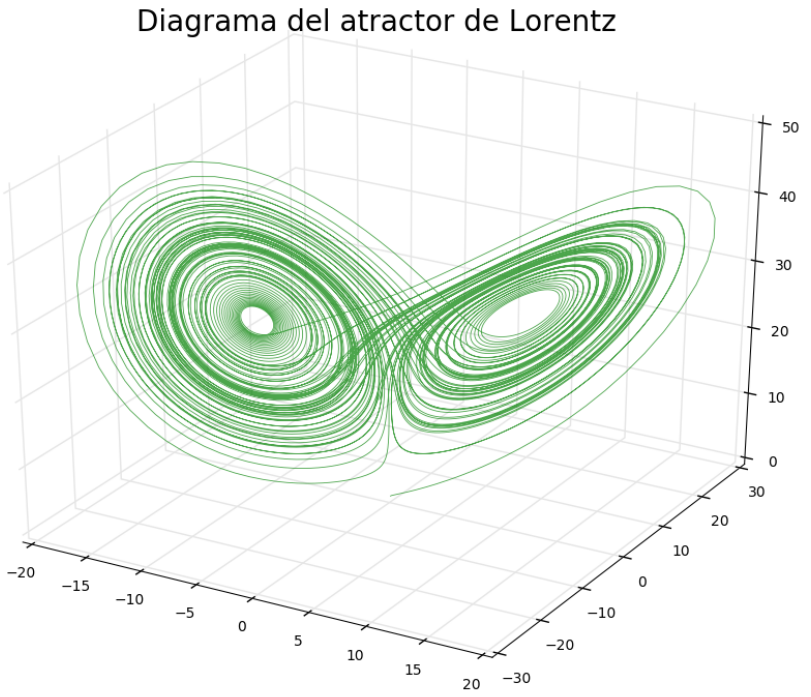


Pensó que se había cometido algún error al ejecutar el programa y lo intentó repetidas veces, logrando siempre el mismo resultado hasta que se dio cuenta de que algo pasaba con el sistema de ecuaciones simplificado con el que estaba trabajando. Después de estudiar detenidamente el problema y hacer pruebas con diferentes parámetros (tanto iniciales como las constantes del sistema), Lorenz llegó a la conclusión de que las simulaciones eran muy diferentes para condiciones iniciales muy próximas. Al llegar a la misma, recordó que en el programa que él había creado para su sistema de meteorología con la computadora Royal McBee, se podían introducir un máximo de 3 decimales para las condiciones iniciales, aunque el programa trabajaba con 6 decimales y los 3 últimos decimales que faltaban se introducían aleatoriamente. Lorenz publicó sus descubrimientos en revistas de meteorología, pasando desapercibidos durante casi una década.

La década de 1970 fue el boom del caos. En 1971 David Ruelle y Floris Takens propusieron una nueva teoría para la turbulencia de fluidos basada en un atractor extraño. Años después el ecólogo teórico Robert May en 1976 encontró ejemplos de caos en dinámica de poblaciones usando la ecuación logística discreta. A continuación llegó el más sorprendente descubrimiento de todos de la mano de Feigenbaum. Él descubrió que hay un conjunto de leyes universales concretas que diferencian la transición entre el comportamiento regular y el caos, por tanto, es posible que dos sistemas evolucionen hacia un comportamiento caótico igual.

Atractor de Lorentz en Jupyter Notebook

Utilizando los comandos proporcionados por gboeing/lorenz-system en github.com obtenemos el siguiente diagrama del atractor de Lorenz con los parametros $\sigma = 10, \rho = 28, \beta = \frac{8}{3}$



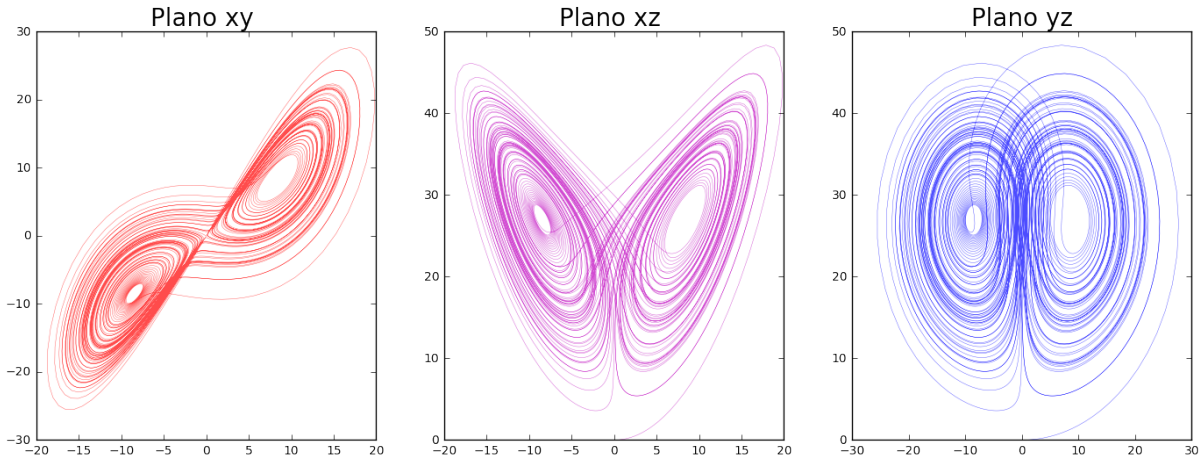
Esto se obtiene utilizando los códigos mostrados a continuación

```
%matplotlib inline
import numpy as np, matplotlib.pyplot as plt,
matplotlib.font_manager as fm, os
from scipy.integrate import odeint
from mpl_toolkits.mplot3d.axes3d import Axes3D

font_family = 'Myriad Pro'
title_font = fm.FontProperties(family=font_family, style='normal', size=20,
weight='normal', stretch='normal')
```

```
save_folder = 'images'
if not os.path.exists(save_folder):
os.makedirs(save_folder)

# define the initial system state (aka x, y, z positions in space)
initial_state = [0.1, 0, 0]

# define the system parameters sigma, rho, and beta
sigma = 10.
rho   = 28.
beta  = 8./3.

# define the time points to solve for, evenly spaced between the start
and end times
start_time = 0
end_time = 100
time_points = np.linspace(start_time, end_time, end_time*100)

# use odeint() to solve a system of ordinary differential equations
# the arguments are:
# 1, a function - computes the derivatives
# 2, a vector of initial system conditions (aka x, y, z positions in space)
# 3, a sequence of time points to solve for
# returns an array of x, y, and z value arrays for each time point, with
the initial values in the first row
xyz = odeint(lorenz_system, initial_state, time_points)

# extract the individual arrays of x, y, and z values from the array of arrays
x = xyz[:, 0]
y = xyz[:, 1]
z = xyz[:, 2]

# plot the lorenz attractor in three-dimensional phase space
fig = plt.figure(figsize=(12, 9))
ax = fig.gca(projection='3d')
ax.xaxis.set_pane_color((1,1,1,1))
ax.yaxis.set_pane_color((1,1,1,1))
ax.zaxis.set_pane_color((1,1,1,1))
```

```
ax.plot(x, y, z, color='b', alpha=0.7, linewidth=0.6)
ax.set_title('Diagrama del atractor de Lorentz', fontproperties=title_font)

fig.savefig('{}/lorenz-attractor-3d.png'.format(save_folder), dpi=180,
bbox_inches='tight')
plt.show()
```

Viendo el esquema desde otra perspectiva, obtenemos los siguientes puntos de vista



Utilizando los siguientes códigos

```
# now plot two-dimensional cuts of the three-dimensional phase space
fig, ax = plt.subplots(1, 3, sharex=False, sharey=False, figsize=(17, 6))

# plot the x values vs the y values
ax[0].plot(x, y, color='r', alpha=0.7, linewidth=0.3)
ax[0].set_title('Plano xy ', fontproperties=title_font)

# plot the x values vs the z values
ax[1].plot(x, z, color='m', alpha=0.7, linewidth=0.3)
ax[1].set_title('Plano xz ', fontproperties=title_font)

# plot the y values vs the z values
```

```
ax[2].plot(y, z, color='b', alpha=0.7, linewidth=0.3)
ax[2].set_title('Plano yz', fontproperties=title_font)

fig.savefig('{}/lorenz-attractor-phase-plane.png'.format(save_folder), dpi=180
, bbox_inches='tight')
plt.show()
```

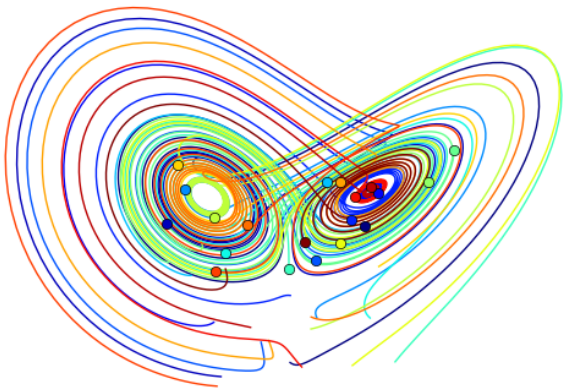
Animación del atractor de Lorenz

Utilizando el código mostrado a continuación podremos obtener una animación del atractor de Lorentz

```
import numpy as np
from scipy import integrate

from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import cnames
from matplotlib import animation

N_trajectories = 20


def lorentz_deriv(params, t0, sigma=10., beta=8./3, rho=28.0):
    """Compute the time-derivative of a Lorentz system."""
    x, y, z = params
    return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

# Choose random starting points, uniformly distributed from -15 to 15
np.random.seed(1)
x0 = -15 + 30 * np.random.random((N_trajectories, 3))

# Solve for the trajectories
t = np.linspace(0, 4, 1000)
x_t = np.asarray([integrate.odeint(lorentz_deriv, x0i, t)
                  for x0i in x0])
```

```
# Set up figure & 3D axis for animation
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1], projection='3d')
ax.axis('off')

# choose a different color for each trajectory
colors = plt.cm.jet(np.linspace(0, 1, N_trajectories))

# set up lines and points
lines = [ax.plot([], [], [], '-', c=c)[0]
for c in colors]
pts = [ax.plot([], [], [], 'o', c=c)[0]
for c in colors]

# prepare the axes limits
ax.set_xlim((-25, 25))
ax.set_ylim((-35, 35))
ax.set_zlim((5, 55))

# set point-of-view: specified by (altitude degrees, azimuth degrees)
ax.view_init(30, 0)

# initialization function: plot the background of each frame
def init():
    for line, pt in zip(lines, pts):
        line.set_data([], [])
        line.set_3d_properties([])

        pt.set_data([], [])
        pt.set_3d_properties([])
    return lines + pts

# animation function.  This will be called sequentially with the frame number
def animate(i):
    # we'll step two time-steps per frame.  This leads to nice results.
    i = (2 * i) % x_t.shape[1]

    for line, pt, xi in zip(lines, pts, x_t):
```

```
            x, y, z = xi[:i].T
            line.set_data(x, y)
            line.set_3d_properties(z)

            pt.set_data(x[-1:], y[-1:])
            pt.set_3d_properties(z[-1:])

        ax.view_init(30, 0.3 * i)
        fig.canvas.draw()
        return lines + pts

    # instantiate the animator.
    anim = animation.FuncAnimation(fig, animate, init_func=init,
                                   frames=500, interval=30, blit=True)

    # Save as mp4. This requires mplayer or ffmpeg to be installed
    anim.save('lorentz_attractor.mp4', fps=15, extra_args=['-vcodec', 'libx264'])

    plt.show()
```



Bibliografía

[1] Wikipedia *Teoría del caos* https://en.wikipedia.org/

[2] Geoff Boeing *Lorenz System*, https://github.com/gboeing/lorenz-system/blob/master/lorenz-system-attractor-visualize.ipynb