Universidad de Sonora

División de Ciencias Exactas y Naturales

Física computacional I

Atractores Extraños.

Moreno Chávez Jesús Rodolfo
Profesor: Carlos Lizárraga Celaya

9 de Mayo del 2017

Resumen

En el presente texto se habla sobre el atractor de Lorenz y la relación que tuvo con el surgimiento de la teoría del caos. Además se presenta un código empleado en python para generar gráficas y animaciones del atractor de Lorenz.

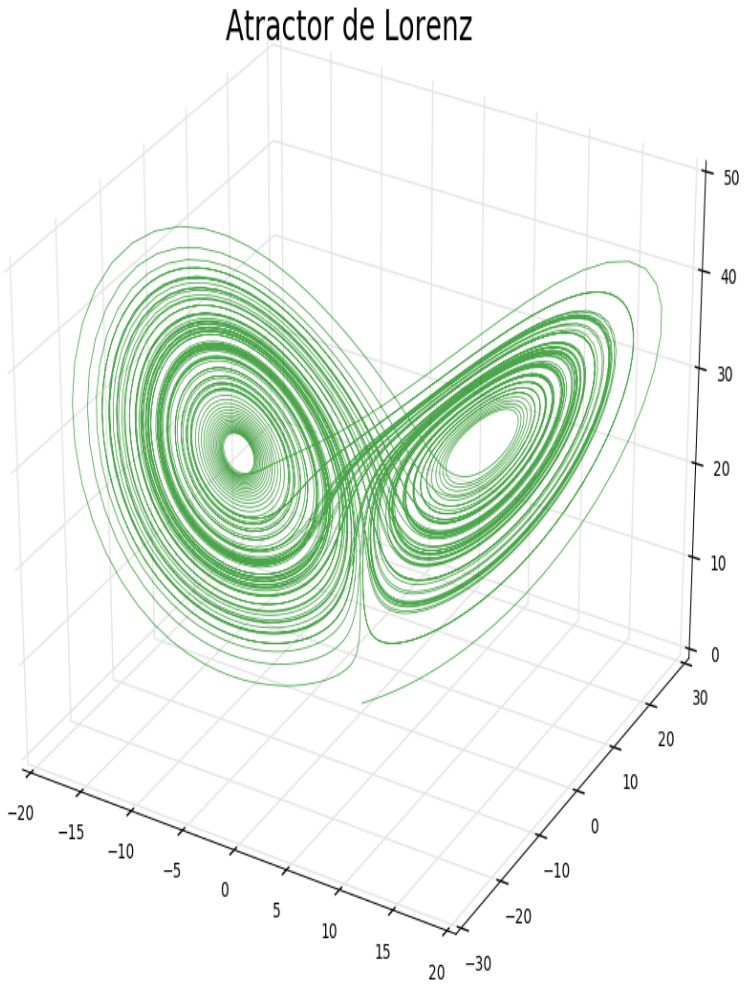
El atractor de Lorenz

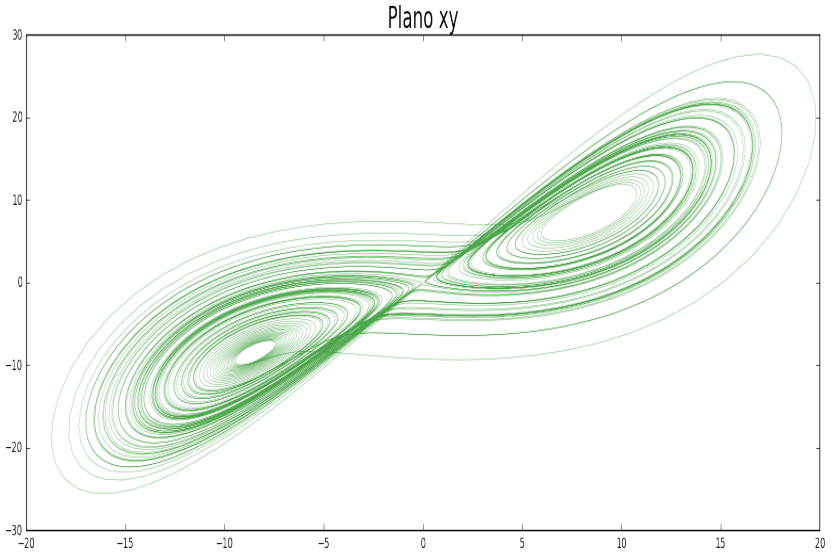
El comienzo de la teoría del caos se sitúa en la década de 1950 cuando se inventaron los ordenadores y se desarrollaron algunas intuiciones sobre el comportamiento de los sistemas no lineales. Esto es, cuando se vieron las primeras gráficas sobre el comportamiento de estos sistemas mediante métodos numéricos.
En 1963, Edward Lorenz(1917-2008), quien se interesaba en el problema de la convección en la atmósfera terrestre, simplificó bastante las ecuaciones de Navier-Stokes de la mecánica de fluidos, conocidas por su complejidad y trató mediante los ordenadores de ver gráficamente el comportamiento de sus ecuaciones.
Lorenz encontró que pequeñas diferencias en un sistema dinámico como la atmósfera terrestre pueden desencadenar un vasto y en muchas ocasiones resultados inesperados.
Las ecuaciones de lorenz, presentadas en su escrito "Deterministic Nonperiodic Flow",son las que dieron pie a formularar lo que hoy es conocida como la teoría del caos.Lorenz derivó este sistema tridimensional de ecuaciones diferenciales no-lineales, sistema que es un modelo matemático simplificado de la recirculación por convección que aparece en la atmósfera.
Lorenz se percató de dos aspectos fundamentales que ocurrían en su modelo: Uno era que alguna diferencia en las condiciones iniciales antes de los cálculos, incluso infinitesimal, cambia de forma drástica los resultados. La predicción solo puede hacerse para periodos de tiempos cortos. Debido a esto hay una extrema sensibilidad a las condiciones iniciales.Otro aspecto es que la impredecibilidad del sistema no implica un comportamiento azaroso, tiene una curiosa tendencia a evolucionar dentro de una zona muy concreta del espacio de fases.
Descubrió que este simple modelo puede desarrollar una dinámica errática extrema: sobre un amplio rango de parámetros, las soluciones oscilan irregularmente, nunca repitiéndose exactamente, pero siempre permaneciendo entre los límites de una región del espacio de fase. Cuando Lorenz graficó las trayectorias en el espacio tridimensional, observó que se situaba en un complicado arreglo, hoy conocido como atractor extraño. El atractor extraño, no es un punto o una curva o una superficie, es un fractal con una dimensión entre 2 y 3.

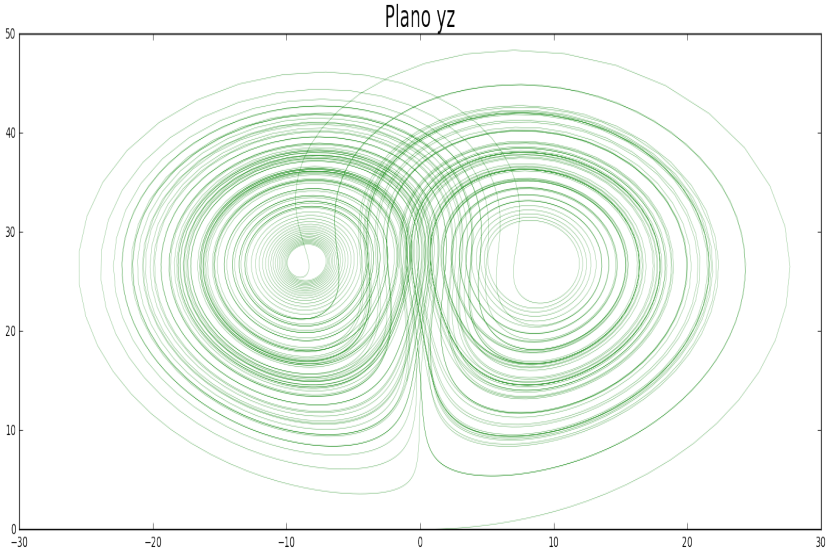
El modelo de lorenz está constituido por el siguiente sistema de ecuaciones:

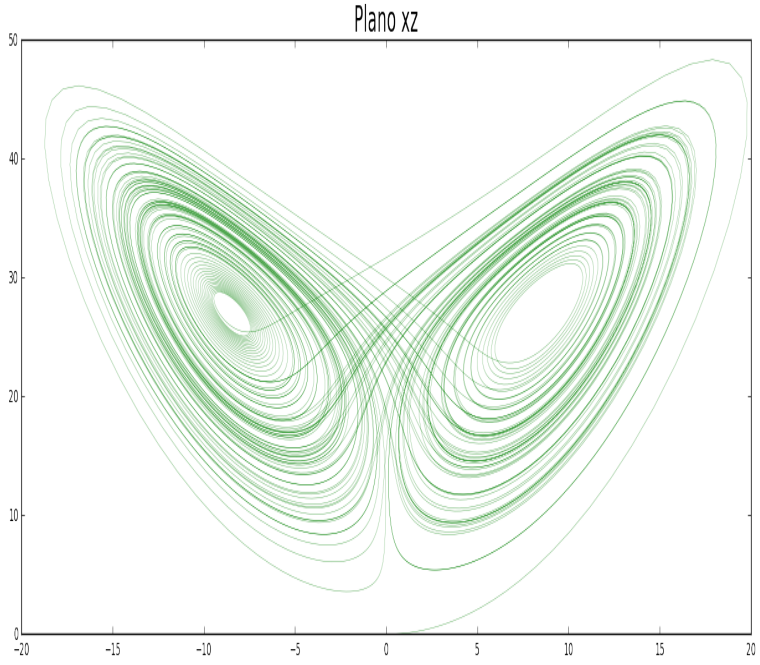
$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned}$$

Para valores de $\sigma = 10$, $\rho = 28$ y $\beta = \frac{8}{3}$, por ejemplo, se obtiene el siguiente atractor:




Plano xy
30
20
10
0
−10
−20
−30
−20
−15
−10
−5
0
5
10
15
20


Plano yz
50
40
30
20
10
0
−30
−20
−10
0
10
20
30



Gráficas y animación del atractor (procedimiento)

Gráficas

Las gráficas del atractor de Lorenz que se mostraron en la sección anterior se realizaron en python con la ayuda de la biblioteca de Matplotlib mediante el siguiente código:

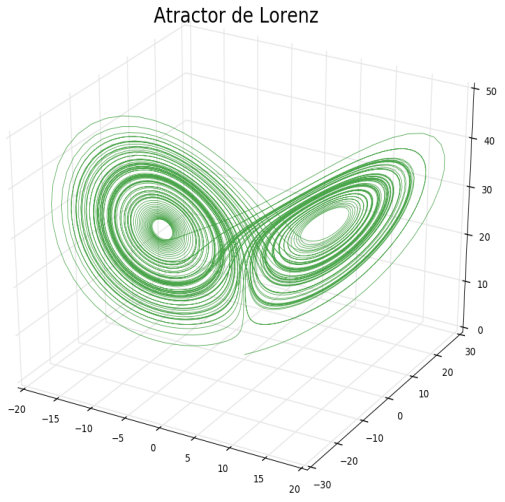
```
%matplotlib inline
import numpy as np, matplotlib.pyplot as plt, matplotlib.font_manager as fm, os
from scipy.integrate import odeint
from mpl_toolkits.mplot3d.axes3d import Axes3D

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

Aquí introducimos el valor de los parámetros σ , ρ y β

```
# 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 time
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 initia
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='green', alpha=0.7, linewidth=0.6)
ax.set_title('Atractor de Lorenz', fontproperties=title_font)

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

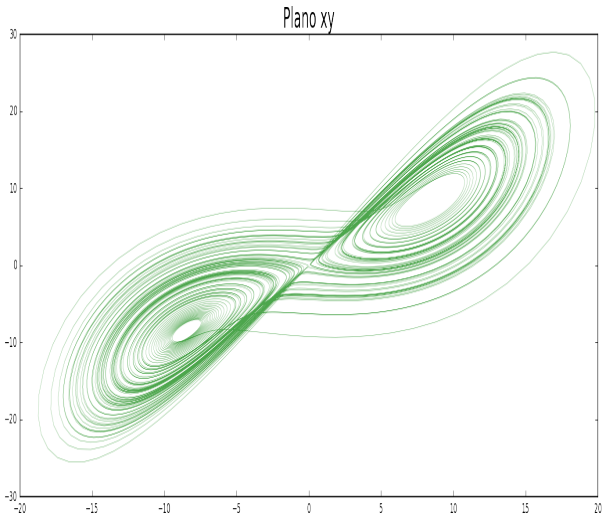
Y como resultado obtenemos la gráfica que presentamos anteriormente:



Para visualizar la gráfica de diversos planos usamos el siguiente código:

```
#Visualizándolo en diversos planos
#Plano x-y
# now plot two-dimensional cuts of the three-dimensional phase space

#Aqui indicamos que plano queremos visualizar
fig, ax = plt.subplots( sharex=False, sharey=False, figsize=(17, 6))
# plot the x values vs the y values
ax.plot(x, y, color='green', alpha=0.7, linewidth=0.3)
ax.set_title('Plano xy', fontproperties=title_font)
plt.show()
```

Y obtenemos lo siguiente

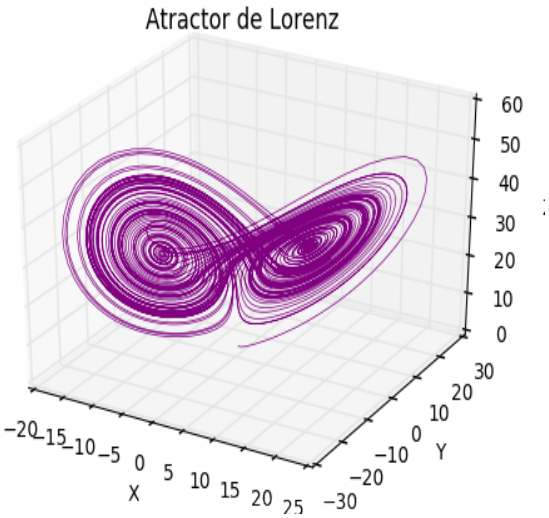


Otra variante para realizar una graica es mediante el siguiente código:

```
# Plot of the Lorenz Attractor based on Edward Lorenz's 1963 "Deterministic
# Nonperiodic Flow" publication.
# http://journals.ametsoc.org/doi/abs/10.1175/1520-0469%281963%29020%3C0130%3ADNF%3
#
# Note: Because this is a simple non-linear ODE, it would be more easily
#       done using SciPy's ode solver, but this approach depends only
#       upon NumPy.

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def lorenz(x, y, z, s=10, r=28, b=2.667):
    x_dot = s*(y - x)
    y_dot = r*x - y - x*z
    z_dot = x*y - b*z
    return x_dot, y_dot, z_dot


dt = 0.01
stepCnt = 10000

# Need one more for the initial values
```

```
xs = np.empty((stepCnt + 1,))
ys = np.empty((stepCnt + 1,))
zs = np.empty((stepCnt + 1,))

# Setting initial values
xs[0], ys[0], zs[0] = (0., 1., 1.05)

# Stepping through "time".
for i in range(stepCnt):
    # Derivatives of the X, Y, Z state
    x_dot, y_dot, z_dot = lorenz(xs[i], ys[i], zs[i])
    xs[i + 1] = xs[i] + (x_dot * dt)
    ys[i + 1] = ys[i] + (y_dot * dt)
    zs[i + 1] = zs[i] + (z_dot * dt)

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.plot(xs, ys, zs, lw=0.5,color='purple',)
ax.set_xlabel("X ")
ax.set_ylabel("Y ")
ax.set_zlabel("Z")
ax.set_title("Atractor de Lorenz")

plt.show()
```

Y se obtiene la siguiente gráfica:



Animación

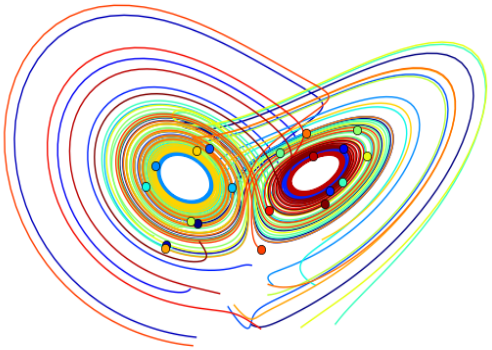
Es posible ralizar animaciones del atractor de Lorenz, a continuación se presenta el código empleado para la animación:

```
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=15., 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)
#guardar un video de la animación como mp4
# Save as mp4. This requires mplayer or ffmpeg to be installed
anim.save('lorentz_attractor3.mp4', fps=15, extra_args=['-vcodec', 'libx264'])
plt.show()
```

En la carpeta de este donde se encuentra este texto se presentan algunas animaciones para distintos valores de los prámetros



Bibliografía

[1] Wikipedia *Teoría del caos*, A 13 de Mayo del 2017, https://en.wikipedia.org/

[2] Animación *Notas de jakevdp* ,https://jakevdp.github.io /blog/2013/02/16/animating-the-lorentz-system-in-3d/

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

[4] example atractor , `https://matplotlib.org/2.0.0/examples/mplot3d/lorenz_attractor.html`

[5] CHAOS, http://www.chaos-math.org/es/caos-vii-atractores-extranosb

[6] E. N. Lorenz, Deterministic Nonperiodic Flow, Journal of the Atmospheric Sciences, vol. 20, p. 130-141, 1963.