Universidad de Sonora

Departamento de ciencias exactas

Física Computacional

El atractor de Lorenz, Ejemplo de caos dinámico

Alumno:
Martha Anahí Iñiguez Beltrán

214202804

April 27, 2018

Contents

1 Introducción

En 1963, Edward Lorenz (1917-2008), quien se interesaba en el problema de la convección en la atmósfera terrestre, simplificó de forma drástica las ecuaciones de Navier-Stokes de la mecánica de fluidos, conocidas por su complejidad. El modelo atmosférico de Lorenz es lo que los físicos llaman un modelo de juguete: aunque probablemente no corresponda con la realidad, Lorenz no tardó mucho en darse cuenta que era un modelo matemático muy interesante. Las ecuaciones de Lorenz dependen de tres números x, y y z, de manera que cada punto del espacio (x,y,z) representa un estado de la atmósfera y para estudiar su evolución hay que seguir un campo de vectores.

2 Código

El código empleado para desarrollar cada unas de las visualizaciones y animaciones queda expuesto a continuación. Nótese que se muestra el código de ejemplo de una sola visualización y una animación.

2.1 Visualización

```
%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=

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

# Definimos el estado inicial del sistema (x, y, z posiciones en el espacio)
initial_state = [0.1, 0, 0]

#Definimos los parametro sigma, rho y beta del sistema
sigma = 10.
```

```
rho   = 28.
beta  = 8./3.

# Definimos los puntos de tiempo a resolver espaciados uniformemente entre tiempo i
start_time = 0
end_time = 100
time_points = np.linspace(start_time, end_time, end_time*100)

# Definimos el sistema de Lorenz
# x, y, yz componen el estado del sistema, t es el tiempo, y sigma, rho, beta son l
def lorenz_system(current_state, t):

    # posiciones de x, y, z en el espacio en el tiempo actual
    x, y, z = current_state

    # definir las 3 ecuaciones diferenciales ordinarias conocidas como las ecuacion
    dx_dt = sigma * (y - x)
    dy_dt = x * (rho - z) - y
    dz_dt = x * y - beta * z

    # devolver una lista de las ecuaciones que describen el sistema
    return [dx_dt, dy_dt, dz_dt]

    # Usa odeint () para resolver un sistema de ecuaciones diferenciales ordinarias
# los argumentos son:
# 1, una función - calcula las derivadas
# 2, un vector de condiciones iniciales del sistema (también conocido como #posicio
# 3, una secuencia de puntos de tiempo para resolver
# devuelve una matriz de matrices de valores x, y y z para cada punto de tiempo, co
xyz = odeint(lorenz_system, initial_state, time_points)

# extraer las matrices individuales de los valores x, y y z de la matriz de matrice
x = xyz[:, 0]
y = xyz[:, 1]
z = xyz[:, 2]

# trazar el atractor Lorenz en el espacio de fase tridimensional
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='g', alpha=0.7, linewidth=0.6)
ax.set_title('Lorenz attractor phase diagram', fontproperties=title_font)

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

# ahora trazar cortes bidimensionales del espacio de fase tridimensional
fig, ax = plt.subplots(1, 3, sharex=False, sharey=False, figsize=(17, 6))

# graficar los valores x vs y
ax[0].plot(x, y, color='r', alpha=0.7, linewidth=0.3)
ax[0].set_title('x-y phase plane', fontproperties=title_font)

# graficar los valores x vs z
ax[1].plot(x, z, color='m', alpha=0.7, linewidth=0.3)
ax[1].set_title('x-z phase plane', fontproperties=title_font)

# graficar los valores y vs z
ax[2].plot(y, z, color='b', alpha=0.7, linewidth=0.3)
ax[2].set_title('y-z phase plane', fontproperties=title_font)

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

# graficar la solución que se generó
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline

figure(1, figsize=(10, 6))

xlabel('t')
grid(True)
hold(True)
lw = 1

plot(time_points, x, 'm', linewidth=lw)
```

```
plot(time_points, y, 'b', linewidth=lw)
plot(time_points, z, 'r', linewidth=lw)

legend((r'$x$', r'$y$', r'$z$'), prop=FontProperties(size=16))
title('Evolución de condición inicial/en función del tiempo')
savefig('Evolución 1.png', dpi=100)
```

2.2 Animación

```
%matplotlib inline
import numpy as np, matplotlib.pyplot as plt, glob, os
import IPython.display as IPdisplay, matplotlib.font_manager as fm
from scipy.integrate import odeint
from mpl_toolkits.mplot3d.axes3d import Axes3D
from PIL import Image

# define las fuentes para usar en las gráficas
family = 'Myriad Pro'
title_font = fm.FontProperties(family=family, style='normal', size=20, weight='norm

save_folder = 'images/lorenz-animate'
if not os.path.exists(save_folder):
    os.makedirs(save_folder)

# definir el estado inicial del sistema (posiciones x, y, z en el espacio)
initial_state = [0.1, 0, 0]

# definir los parametros sigma, rho, y beta del sistema
sigma = 10.
rho   = 28.
beta  = 8./3.

# Definimos los puntos de tiempo a resolver espaciados uniformemente entre tiempo i
start_time = 1
end_time = 60
interval = 100
time_points = np.linspace(start_time, end_time, end_time * interval)
```

```
# definir el sistema de Lorenz
def lorenz_system(current_state, t):
    x, y, z = current_state
    dx_dt = sigma * (y - x)
    dy_dt = x * (rho - z) - y
    dz_dt = x * y - beta * z
    return [dx_dt, dy_dt, dz_dt]

# graficar el sistema en 3 di
def plot_lorenz(xyz, n):
    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))
    x = xyz[:, 0]
    y = xyz[:, 1]
    z = xyz[:, 2]
    ax.plot(x, y, z, color='g', alpha=0.7, linewidth=0.7)
    ax.set_xlim((-30,30))
    ax.set_ylim((-30,30))
    ax.set_zlim((0,50))
    ax.set_title('Lorenz system attractor', fontproperties=title_font)

    plt.savefig('{}/{:03d}.png'.format(save_folder, n), dpi=60, bbox_inches='tight'
    plt.close()

# devolver una lista en trozos iterativamente más grandes
def get_chunks(full_list, size):
    size = max(1, size)
    chunks = [full_list[0:i] for i in range(1, len(full_list) + 1, size)]
    return chunks

# # obtener trozos cada vez más grandes de los puntos de tiempo, para revelar el at
chunks = get_chunks(time_points, size=20)

#obtener los puntos para graficar, un pedazo de tiempo a la vez, integrando el sist
points = [odeint(lorenz_system, initial_state, chunk) for chunk in chunks]

# graficar cada conjunto de puntos, uno a la vez, guardando cada gráfica
```

```
for n, point in enumerate(points):
    plot_lorenz(point, n)

# crear una lista de duraciones de visualización, una para cada fotograma
first_last = 100 #show the first and last frames for 100 ms
standard_duration = 5 #show all other frames for 5 ms
durations = tuple([first_last] + [standard_duration] * (len(points) - 2) + [first_l

# cargar todas las imágenes estáticas en una lista
images = [Image.open(image) for image in glob.glob('{}/*.png'.format(save_folder))]
gif_filepath = 'images/animated-lorenz-attractor.gif'

# guardar como un gif animado
gif = images[0]
gif.info['duration'] = durations #ms per frame
gif.info['loop'] = 0 #how many times to loop (0=infinite)
gif.save(fp=gif_filepath, format='gif', save_all=True, append_images=images[1:])

# verificar que la cantidad de cuadros en el gif sea igual a la cantidad de archivo
Image.open(gif_filepath).n_frames == len(images) == len(durations)

IPdisplay.Image(url=gif_filepath)
```

3 Resultados

A continuacion se muestran los resultados obtenidos de las visualizaciones. El gif será incluido en la carpeta con los archivos generados con nuestro código.

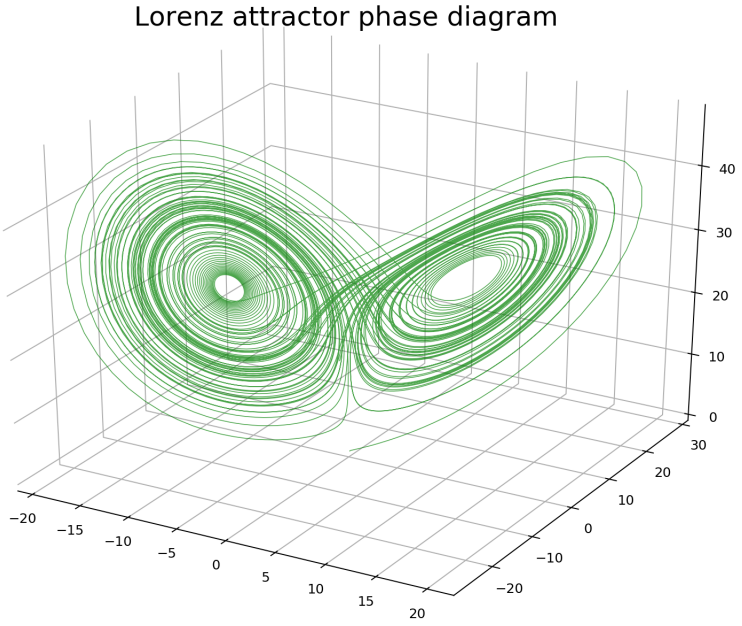

Figure 1: Atractor de Lorenz

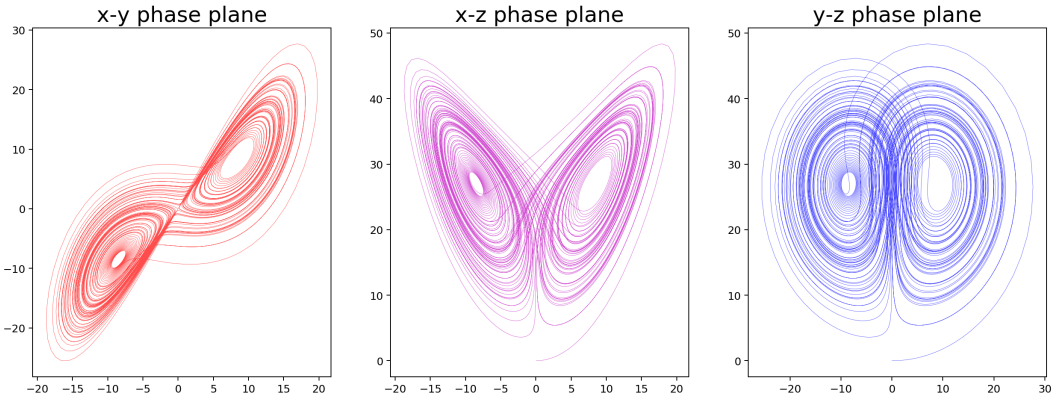

Figure 2: Vista de los planos ortogonales del atractor de 2D

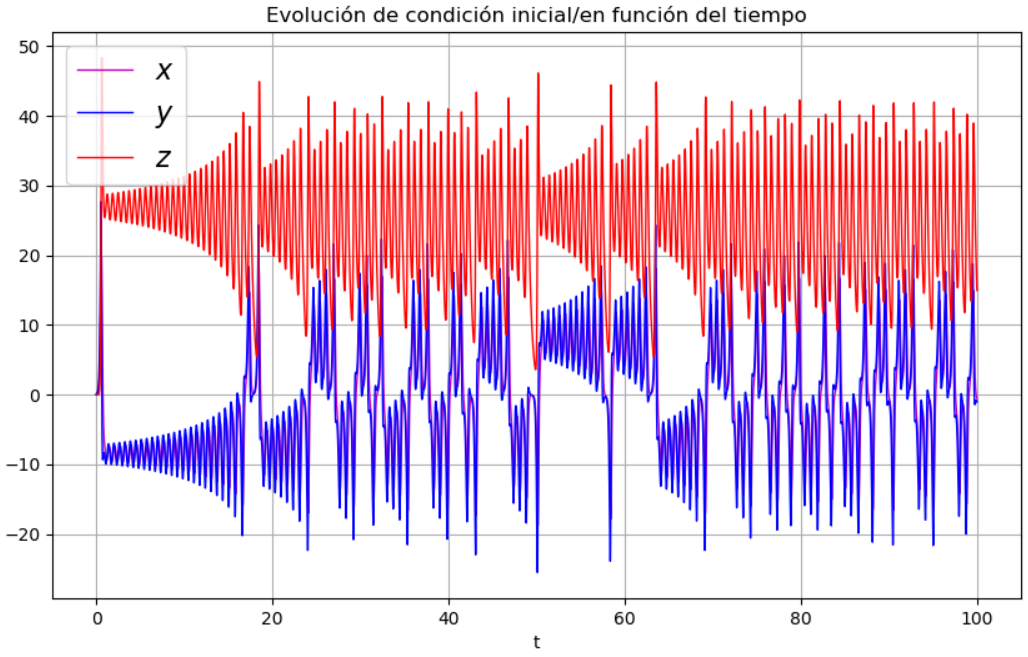

Figure 3: Evolucion de cada variable respecto al tiempo

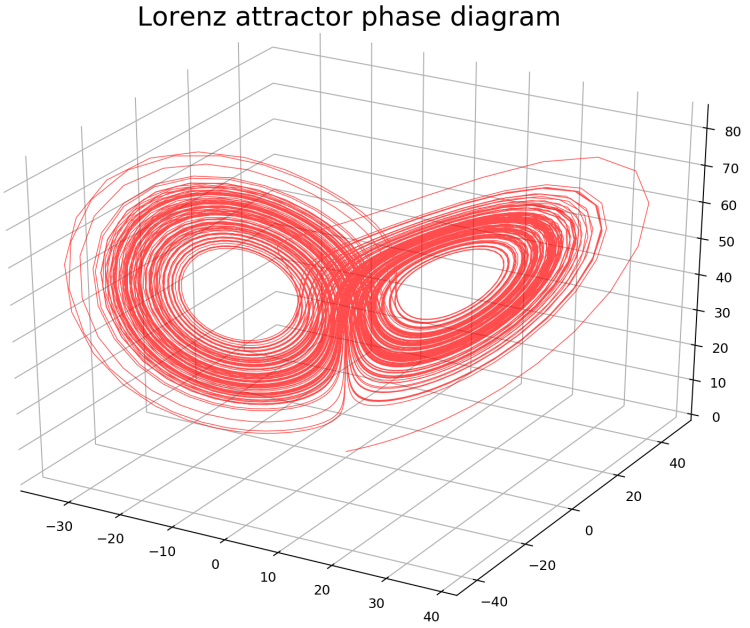

Figure 4: Atractor de Lorenz

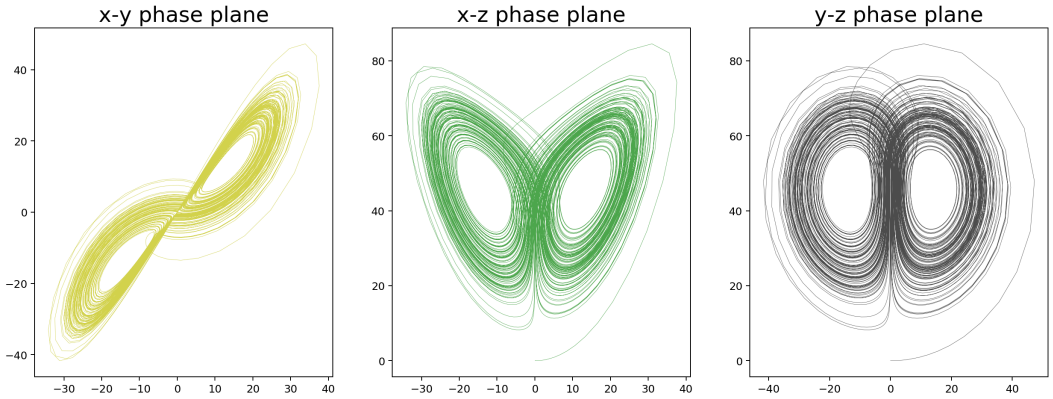

Figure 5: Vista de los planos ortogonales del atractor de 2D

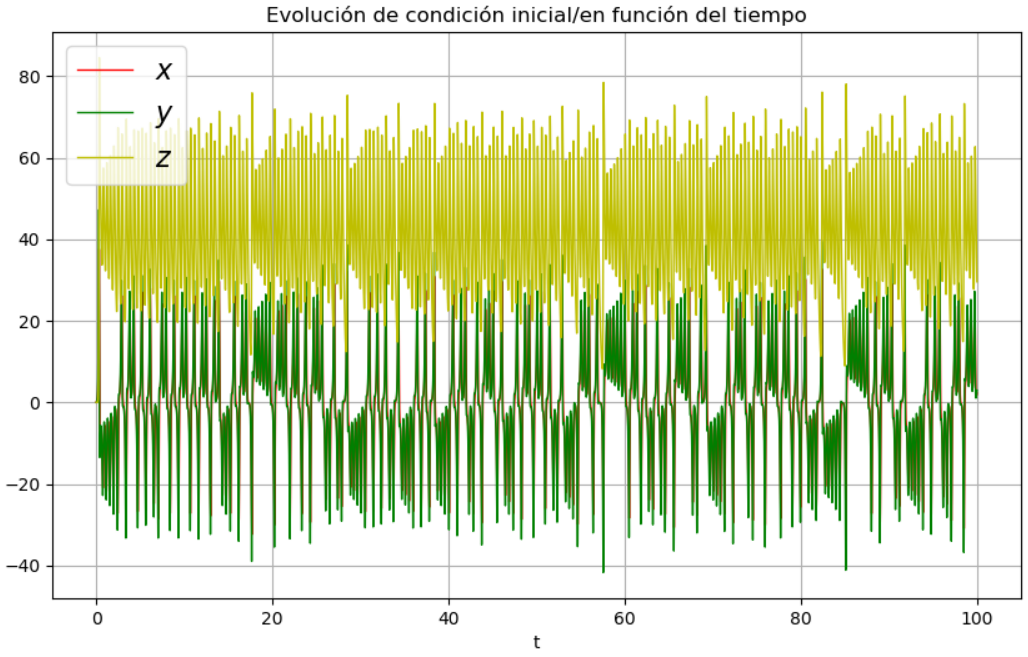

Figure 6: Evolucion de cada variable respecto al tiempo

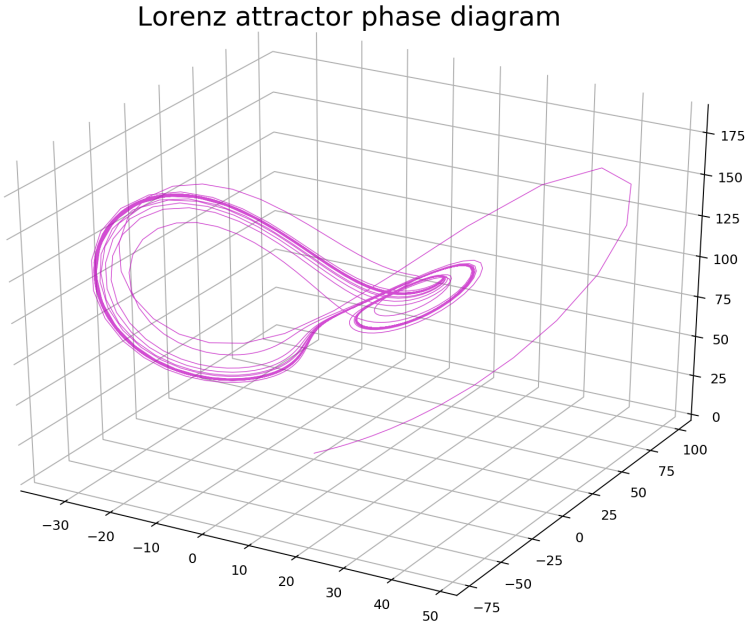

Figure 7: Atractor de Lorenz

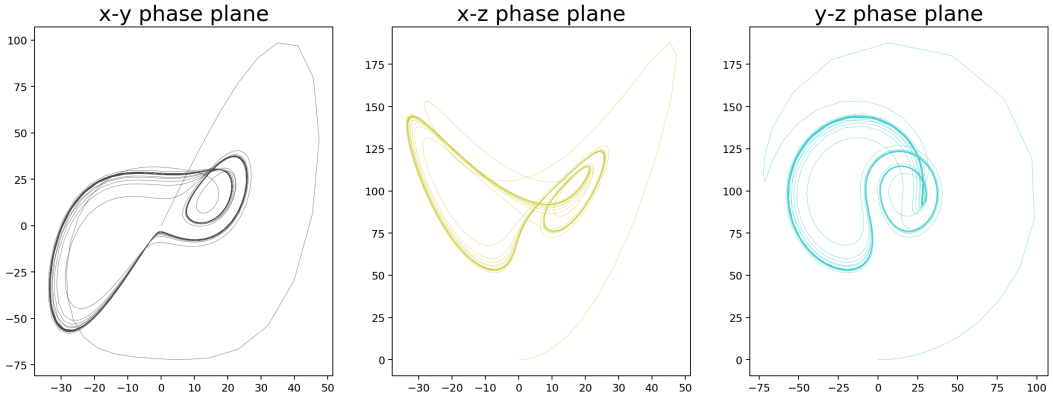

Figure 8: Vista de los planos ortogonales del atractor de 2D

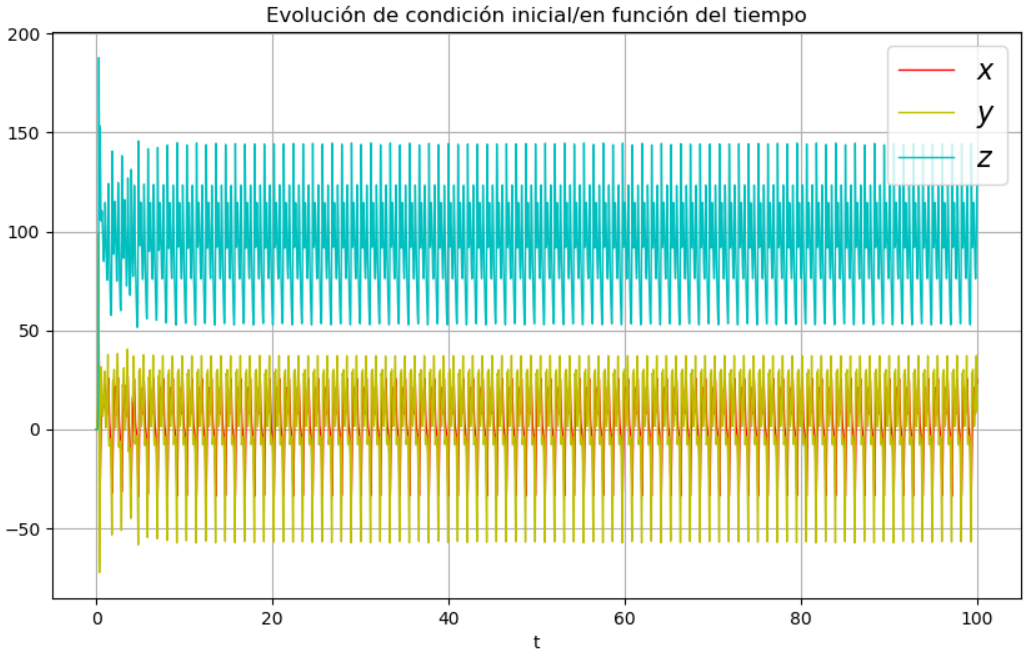

Figure 9: Evolucion de cada variable respecto al tiempo

3.1 Sigma=10, Beta=8/3, Rho=28

3.2 Sigma=28, Beta=4, Rho=46.28

3.3 Sigma=10, Beta=8/3, Rho=99.92

4 Bibliografía

-Atractor de Lorenz(2018) De Wikipedia.org. Recopilado el 26 de Abril del 2018: htt

-J. Leys, E. Ghys, A. Alvarez.(2018) Caos VII: Atractores extraños. Recopilado el 2