Evaluación 2

Jesús Adrián Zatarain Alvarado

April 27, 2018

1 Introducción

En la presente evaluación, se tiene como objetivo el realizar la gráfica y animación del Atractor de Lorentz, un caso del caos dinámico.

Para la actividad se conseguirá un archivo con el código para la visualización y animación del Atractor de Lorentz. Se reproducirá los códigos en cuestión para obtener sus imágenes. Para después modificcar los valores iniciales del concepto matemático y ver las diferencias que resaltan con respecto al original.

Por último, se va a modificar el parámetro "rho" y se va a ve la diferencia que tiene su imagen respecto a las otras.

2 Ejemplo

2.1 Visualización

Primero se importan las bibliotecas a utilizar. ES el caso de numpy, matplotlib, scipy y otros más. Se crea una carpeta donde se van a guardar las imágenes posteriores.

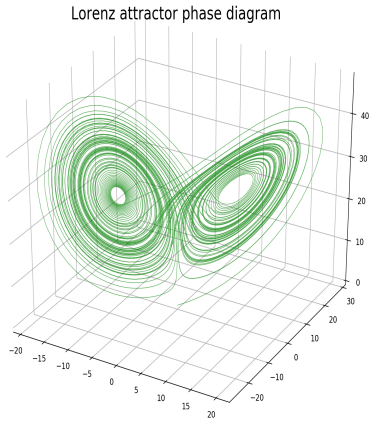
Consiguiente, se definien los parámetros que va a tener el atractor. Las condiciones inciales van a ser 0 para y,z, pero para x va a ser de 0.1. Por otro lado, sigma tendrá un valor de 10, rho de 28 y beta de 8/3.

Después se definieron las tres ecuaciones diferenciales para la realización del atractor:

```
# define the lorenz system
# x, y, and z make up the system state, t is time, and sigma, rho, beta are the system parameters
def lorenz_system(current_state, t):

    # positions of x, y, z in space at the current time point
    x, y, z = current_state

   # define the 3 ordinary differential equations known as the lorenz equations
    dx_dt = sigma * (y - x)
```

```
    dy_dt = x * (rho - z) - y
    dz_dt = x * y - beta * z

    # return a list of the equations that describe the system
    return [dx_dt, dy_dt, dz_dt]
```

Después se resolivió el sistema de ecuaciones, y devolvió un valor para cada variable x, y, z. Para po último llegar a la gráfica del atractor.



2.2 Animación

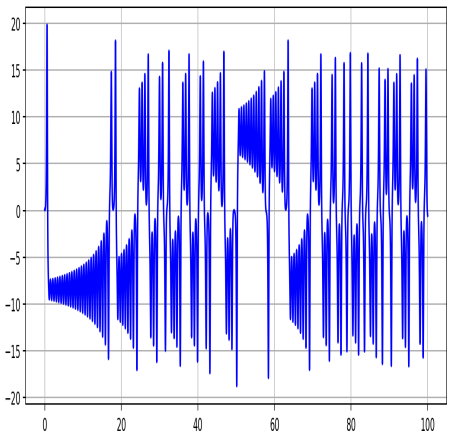
Para esto se utilizaron las mismas condiciones anteriores y el mismo sistema de ecuaciones. Donde se agregó el siguiente codigo para realizar la animación:

Que creará una gran cantidad de imágenes para poder crear la animación.

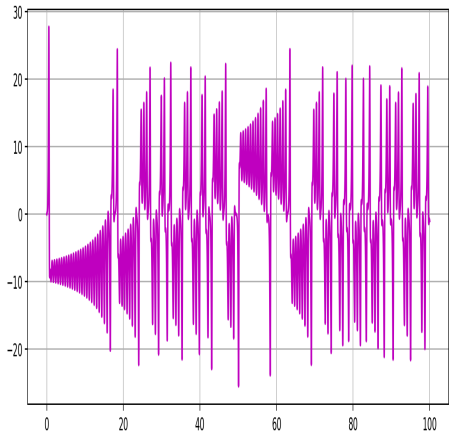
3 Parte 1

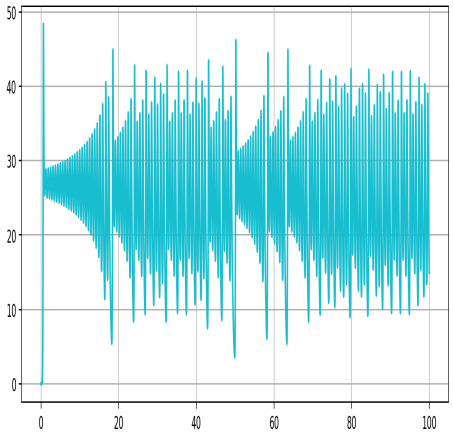
Para esta parte se pidió que se viera el comportamiento de x, y, z respecto al tiempo- Para ello se creó una primera imagen de x-t:

```
from pylab import figure, plot, ylabel, xlabel, grid, hold, legend, title, savefig, ylim, xlim,
figure(figsize=(10, 4))
grid(True)
#hold(True)
lw = 2
plot(time_points, x, 'b', linewidth=lw)

savefig('{}/ox-t.png'.format(save_folder), dpi=180, bbox_inches='tight')
show()
```

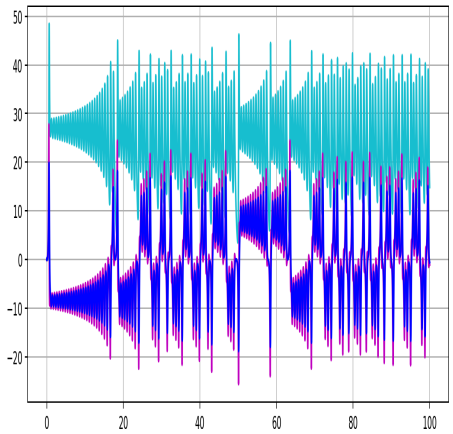


Para las otras comparaciones se sustituyó en el código anterior por y y z, se crearón las siguientes imágenes:





Por último se hizo una comparando las tres directamente:

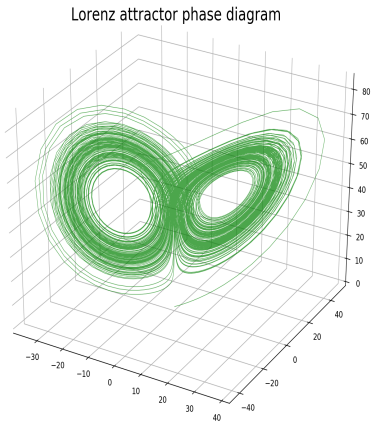


4 Parte 2

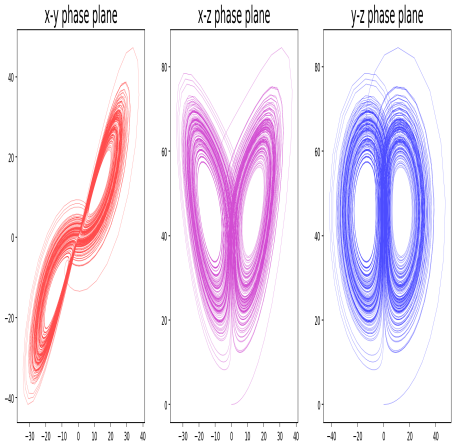
Para esta parte se cambian las condiciones iniciales como sigue:

```
# 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 = 28.
rho   = 46.92
beta  = 4.

# 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)
```

Que crea la siguiente imagen:

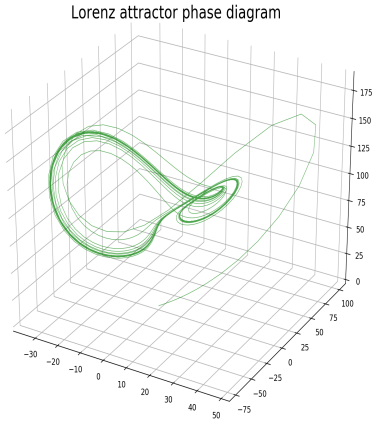


Posterior, se pide crear una comparación de las gráficas respecto a dos ejes:



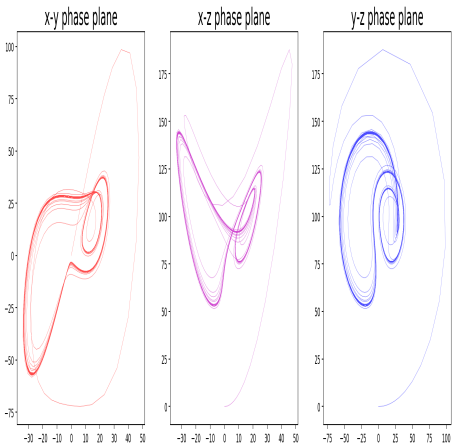
5 Parte 3

Para esta parte se vuelven al valor de los parámetros anteriores y se cambia rho por 99.96.



Se pide analizar el comportamiento de esta nueva imagen:

```
# 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('x-y phase plane', 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('x-z phase plane', 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('y-z phase plane', fontproperties=title_font)

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



En este nuevo ejemplo, se nota un decrecimiento en el tamaño de las longitudes del Atractor