
Universidad de Sonora
División de Ciencias Exactas y Naturales
Departamento de Física

---

Física Computacional I

# El Atractor de Lorenz, ejemplo de Caos Dinámico

---

**Evaluación 2 (2018-1)**

Rolando Abdel Fimbres Grijalva

26 de abril de 2018

# 1. Introducción

El atractor de Lorenz es un sistema dinámico determinista tridimensional no lineal que se derivo de ecuaciones diferenciales que describían el comportamciento del viento así como transferencias de calor producidas en la atmósfera, inicialmente como un modelo matemático simplificado para la convección atmosférica.

# 2. Desarollo

La práctica consta de cuatro pasos:

- Replicar el código proporcionado (del autor Geoff Boeing) para la visualización y animación del atractor.

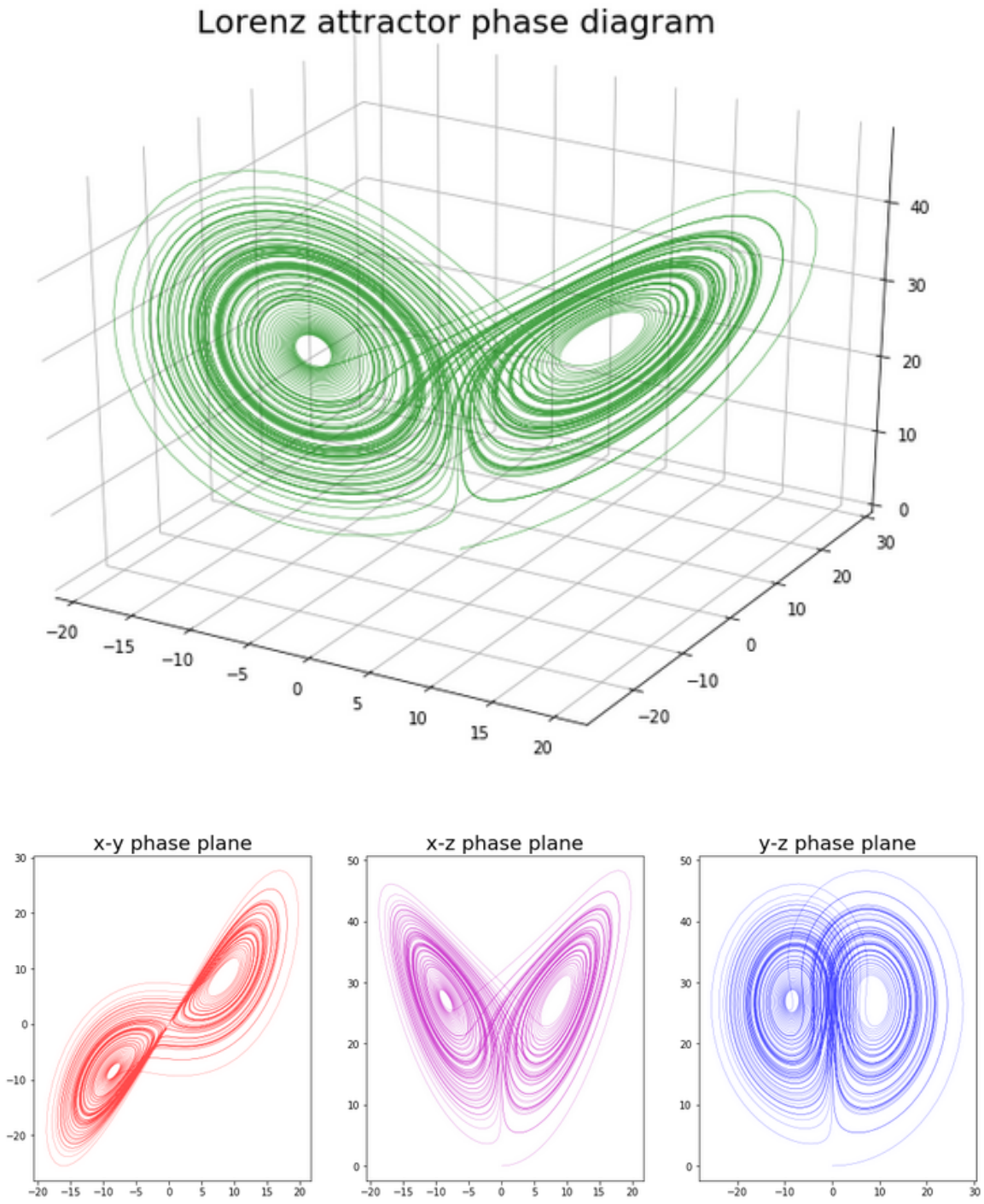
- Construir una gráfica bidimensional que muestre las soluciones de $x(t)$, $y(t)$ y $z(t)$ para así observar la evolución temporal.
- Repetir los mismos cálculos (visualización y animación) pero ahora para: $\sigma = 28$, $\beta = 4$ y $\rho = 46{,}92$ contrastando los casos anteriores.
- Explorar el caso $\sigma = 10$, $\beta = \frac{8}{3}$ y $\rho = 99{,}96$ describiendo los resultados obtenidos.

# 3. Resultados

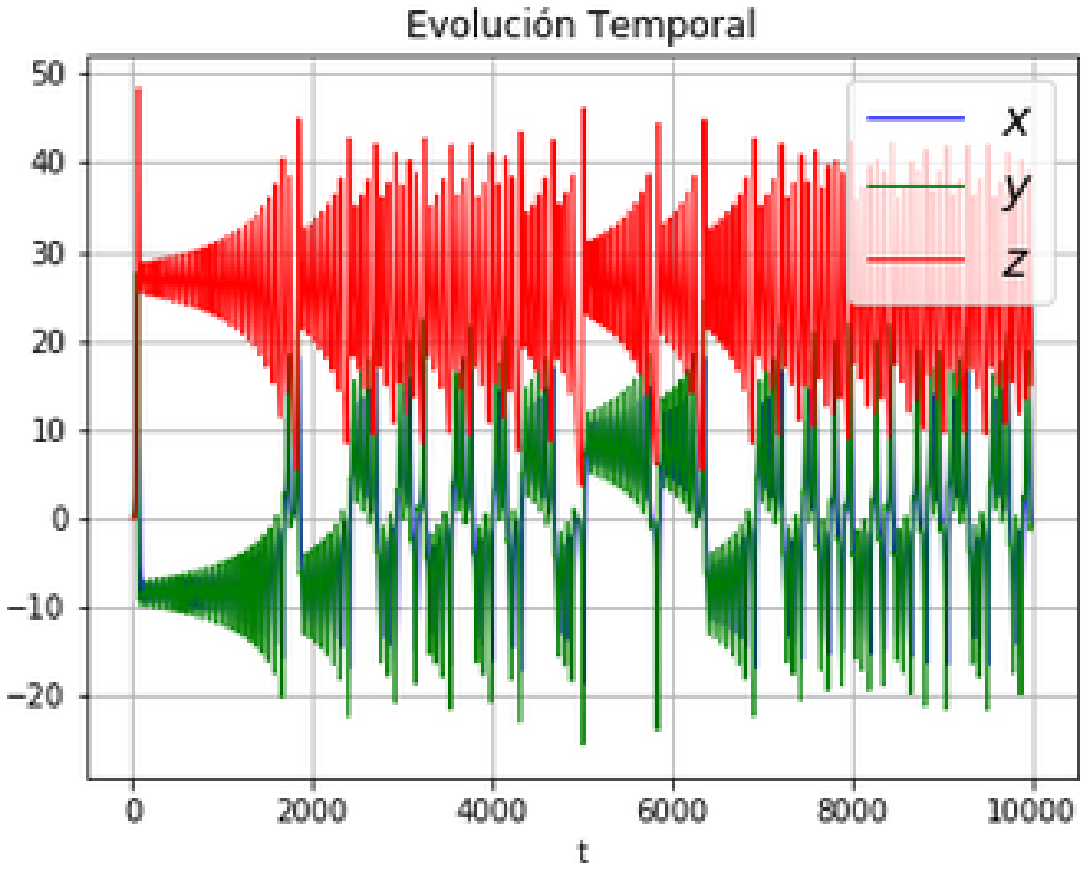
Durante el primer paso replicamos el código e introducimos los valores por defecto para cada variable, obteniendo:

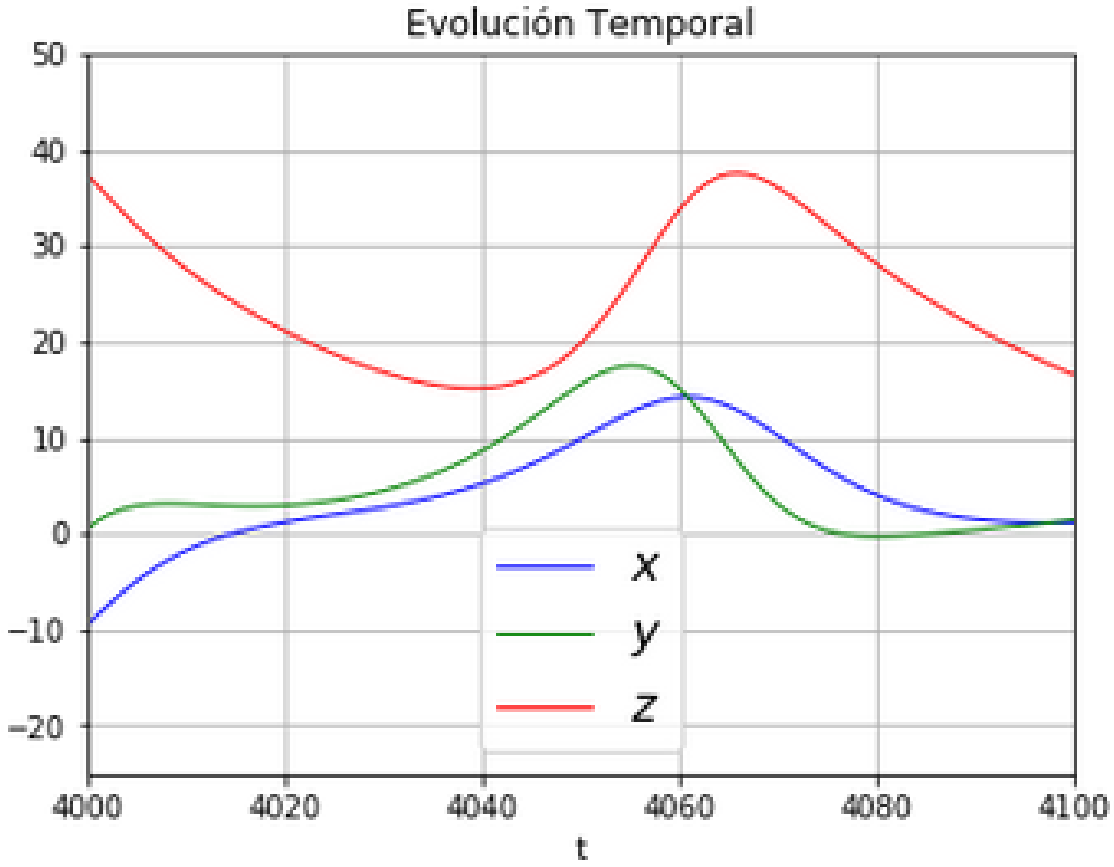
```
# define the initial system state (aka x, y, z positions in space)
initial_state = [0.1, 0, 0]

# 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)
```

```
# define the lorenz system
# x, y, and z make up the system state, t is time, and sigma, rho, beta are the system par
ameters
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]
```



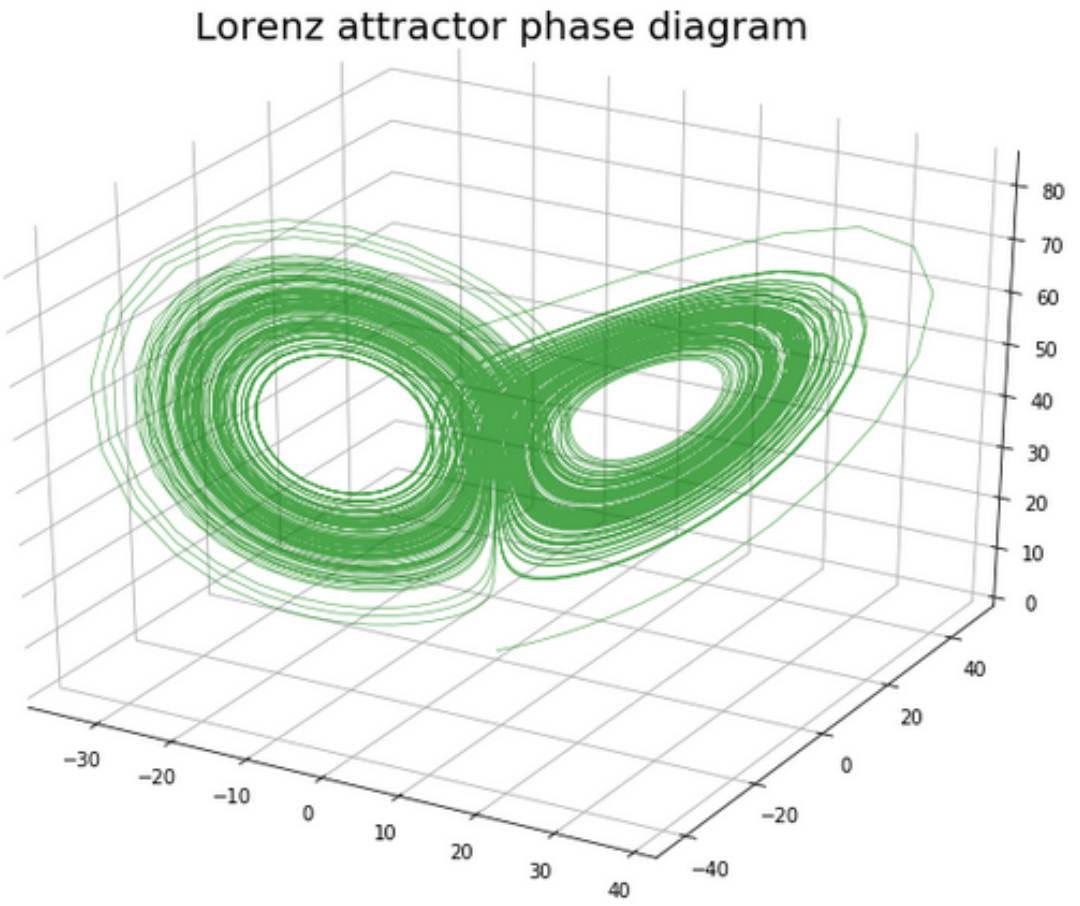

En el segundo caso observamos dos imagenes de la evolución temporal, la segunda en un rango distinto a la anterior.

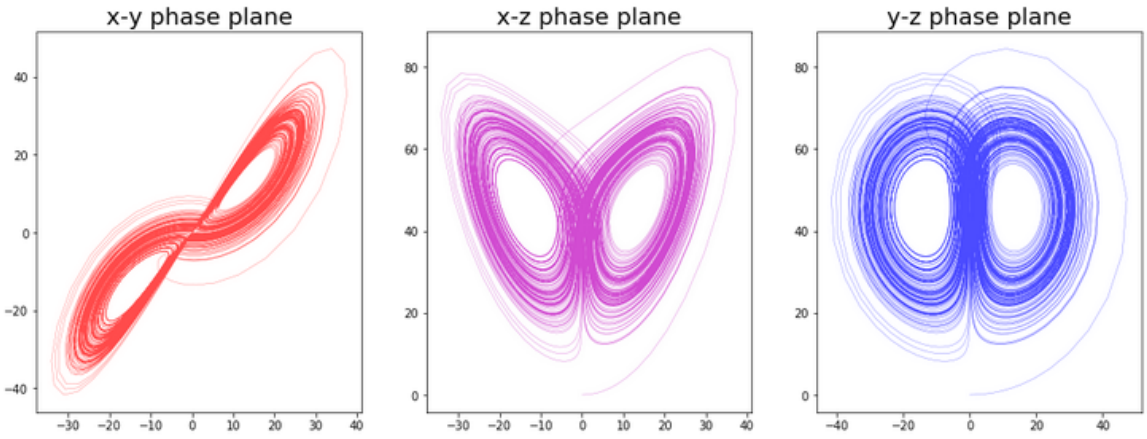

Evolución Temporal
50
40
30
20
10
0
−10
−20
0
2000
4000
6000
8000
10000
t
x
y
z




Para el siguiente caso necesitamos cambiar los parámetros, resultando en:

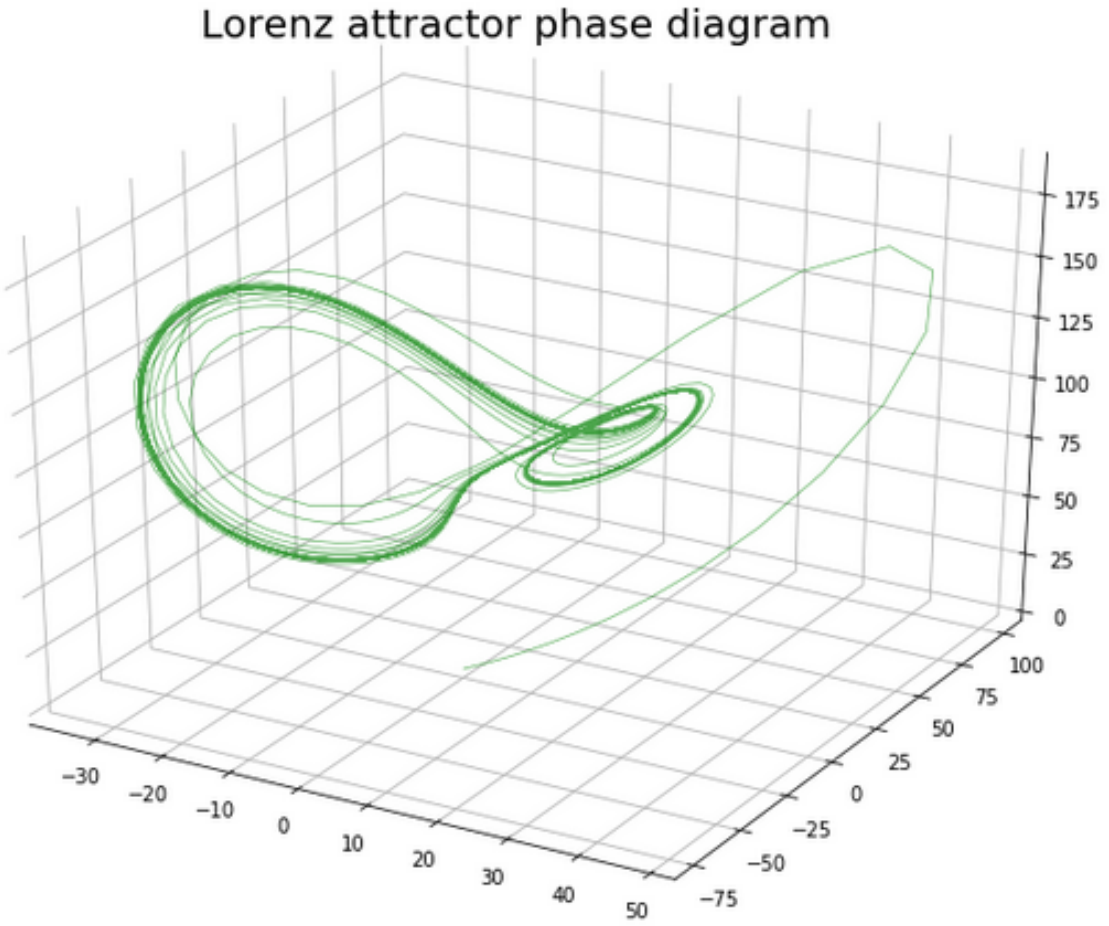
```
# define the initial system state (aka x, y, z positions in space)
initial_state = [0.1, 0, 0]

# define the system parameters sigma, rho, and beta
sigma = 28.0
rho   = 46.92
beta  = 4.0

# 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)
```

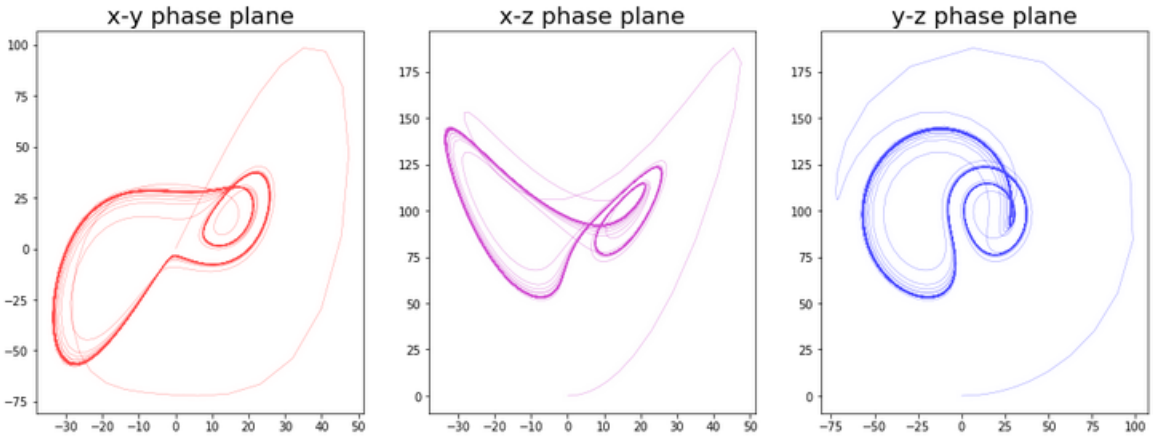



Finalizando con una última modificación a los parámetros.

```
# 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   = 99.96
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)
```






## 3.1. Contraste

En el tercer paso observamos que el cambio ocasionó la concentración del movimiento hacia las orillas del atractor. En cambio en la última modificación observamos un comportamiento completamente distinto a los anteriores, como si se tratase de un fenómeno de otra naturaleza. Observamos que los parámetros influyen en la saturación del atractor algo que notamos facilmente en los diagramas de los distintos planos.