Atractores caóticos en Python

Autores del artículo: Floridia, Micaela - Arrasco, Luna Nicole.
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico.

RESUMEN

El presente trabajo muestra los resultados de diversas simulaciones, programadas en Python, del atractor de Lorenz y de Rössler. El objetivo es visualizar el comportamiento caótico de los sistemas, modificando las condiciones iniciales de sus variables y sus parámetros, y evidenciar las ventajas del uso de aplicaciones de programación básica como herramientas de resolución de sistemas de ecuaciones no lineales.

Para la integración numérica de los sistemas se utilizó el método de Euler. Además, se procedió a realizar los diagramas de bifurcación.

Palabras clave: Atractor de Lorenz - Atractor de Rössler - Método de Euler - Caos- Diagrama de bifurcación.

INTRODUCCIÓN

Los científicos pasan horas de su vida intentando recrear modelos que presenten la realidad para ser estudiados y lograr mejorar las condiciones de vida de las personas. ¿Qué sucede cuando las ecuaciones que describen los problemas tienen un comportamiento impredecible?

La presencia del caos determinista comienza alrededor de los 60 pero su momento de auge se produce en la década de los 70. Se presenta a partir de sistemas dinámicos descritos por ecuaciones no lineales, cuya característica llamativa es la sensibilidad en las condiciones iniciales. Esto quiere decir que, a raíz de una pequeña modificación en las condiciones

iniciales, el resultado se ve afectado radicalmente provocando un comportamiento impredecible. Este fenómeno recibe el nombre de "efecto mariposa".

En las últimas décadas, estos sistemas fueron de gran interés por su aplicación en una variedad de ramas como la física, la ingeniería, la ecología, la biología, etc.

Mientras trabajaba en un modelo simplificado de la confección atmosférica, Edward Lorenz (1917-2008) se encontró con un Atractor extraño en forma de mariposa. El meteorólogo ingresó, en varias oportunidades, las ecuaciones en un ordenador. En una de ellas, utilizó los valores que se habían generado a mitad de una simulación anterior como condiciones iniciales. Se llevó una gran sorpresa al visualizar que, luego de un cierto tiempo, los valores de la segunda parte de la simulación anterior comenzaban a diferir con la primera parte de la simulación ejecutada, hasta no parecerse contradiciendo lo previsto. En otro intento, decidió redondear los valores ingresando únicamente tres cifras decimales en vez de las seis permitidas por el ordenador. En esta oportunidad, se encontró con que pequeñas diferencias en las condiciones iniciales llevaban a una gran diferencia en el pronóstico en relativamente poco tiempo, rompiendo con el paradigma establecido por los físicos hasta ese momento. Frente a estas situaciones y retomando las ideas de Poincaré (1854-1912), introdujo el concepto de sensibilidad a las variaciones en las condiciones iniciales en este tipo de sistemas que provocan que los comportamientos sean impredecibles a largo plazo.

El atractor de Lorenz se modeliza a través de un sistema de tres ecuaciones diferenciales no lineales, que contiene a las derivadas temporales x, y y z, y cuyos parámetros son: σ, ρ y β.

$$\begin{cases} \dot{x} = \sigma.(y - x) \\ \dot{y} = \rho.x - y - xz \\ \dot{z} = xy - \beta z \end{cases}$$

Ecuaciones del Atractor de Lorenz.

En este sistema se visualizan los siguientes parámetros o variables: x representa la razón de rotación del sistema, y el gradiente de temperatura, z la desviación de temperatura, σ el número de Prandtl, ρ el número de Rayleigh y β la razón entre la longitud y la altura del sistema.

Otro ejemplo en donde se hace presente el caos es en el atractor de Rössler. Está formado por un sistema de tres ecuaciones diferenciales ordinarias no lineales, que fueron estudiadas por Otto E. Rössler. Las variables representan la concentración de alguna especie química.

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + by \\ \dot{z} = b + z.(x - c) \end{cases}$$

Ecuaciones del Atractor de Rossler.

En el siguiente trabajo se representarán los atractores mencionados, a través del uso del lenguaje de programación Python.

PROCEDIMIENTOS

El trabajo está dividido en dos secciones, cada una dedicada a un atractor.

Atractor de Lorenz

En un primer momento, se desarrolló el código utilizando funciones definidas por las variables (x,y,z) y los parámetros (σ, ρ, β). A estos últimos se les asignaron sus valores característicos: σ=10, ρ=28 y β=8/3. Luego, se integró numéricamente el sistema utilizando el método de Euler, con un diferencial de tiempo de 0,01 segundos.

Al definir las condiciones iniciales de las variables se utilizaron dos ternas diferentes, modificando solamente el valor inicial de una de ellas, con el fin de poder apreciar gráficamente el resultado al variarlas levemente. Los valores utilizados fueron: (x,y,z)=(1;0;0) y (x2,y2,z2)=(1,1;0;0). Por último, se plasmaron los resultados en un gráfico

en tres dimensiones (figura 1) y se representaron también las evoluciones temporales de las tres variables (figuras 2, 3 y 4).

Al trabajar con funciones, las gráficas resultan grotescas, no permitiendo apreciar de una manera significativa las trayectorias. Por lo tanto, se decidió desarrollar el código repitiendo los mismos pasos anteriores, pero está vez utilizando vectores, obteniendo así las gráficas de las figuras 5, 6, 7 y 8. En este caso, adicionalmente, se representaron tres diferentes cortes: para el plano xy (figura 9), xz (figura 10) e yz (figura 11).

Atractor de Rössler

Para este atractor, se desarrolló directamente el código mediante vectores. Para los parámetros a,b y c se utilizaron sus valores característicos: a=b=0,2 y c=5. Se integró el sistema numéricamente a través del método de Euler, con un diferencial de tiempo de 0.01 segundos. En este caso, se trabajó también con dos ternas diferentes de coordenadas iniciales: (x,y,z)=(1;0;0) y (x2,y2,z2)=(1,1;0;0).

Se graficó el atractor en tres dimensiones (figura 12) y los tres diferentes cortes: para el plano xy (figura 13), para el xz (figura 14) y el yz (figura 15). Se realizaron las gráficas de las evoluciones temporales de las tres variables (figuras 16, 17 y 18).

Por otro lado, para visualizar de qué manera afectan los parámetros a la evolución del sistema, se obtuvieron las soluciones del mismo variando c. Los valores escogidos fueron: c=2,5 (figuras 19 y 20); c=3,5 (figuras 21 y 22), c=4 (figuras 23 y 24) y c=5 (figuras 25 y 26). La decisión de variar el parámetro c se debió a que los resultados obtenidos son más sencillos de analizar.

Para finalizar, se elaboraron dos diagramas de bifurcación variando los parámetros b y c para ver cómo se comportan los puntos fijos del sistema.

RESULTADOS

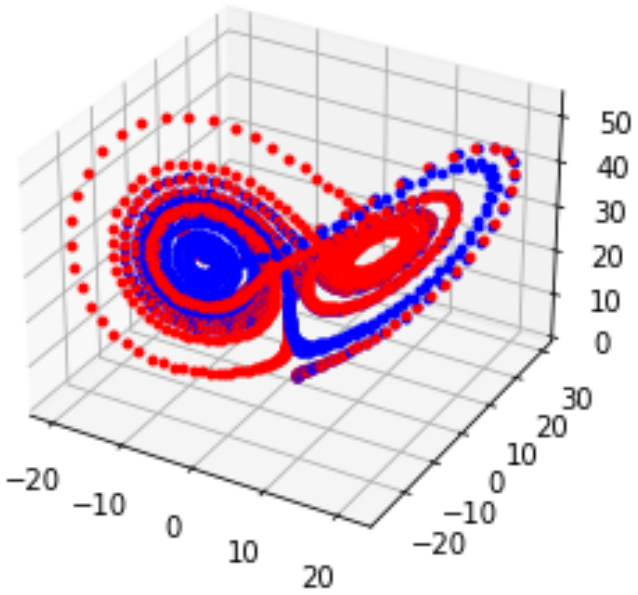
En esta sección se presentarán las diferentes gráficas mencionadas previamente en el procedimiento.

Atractor de Lorenz

Primera parte: Gráficas a partir del código con funciones, por el método de Euler.

En el gráfico 3d se visualizan dos espirales dobles que corresponden a los sistemas generados por diferentes condiciones iniciales. La roja representa al sistema determinado por (x,y,z)=(1;0;0) y la azul a (x2,y2,z2)=(1,1;0;0).

Figura 1: Atractor de Lorenz



Se observa que las trayectorias oscilan entre dos focos formando la silueta de una mariposa.

Como habíamos mencionado anteriormente y gracias a la gráfica, podemos afirmar que una perturbación en las condiciones iniciales, por más pequeña que sea, provoca que las curvas pasen por lugares completamente distintos.

En la serie temporal de x se muestra que la evolución es oscilatoria; pero, si bien parecería tener un periodo, si comparamos los puntos máximos (picos) de la gráfica, concluimos que no hay una periodicidad.

Figura 2: evolución temporal de x

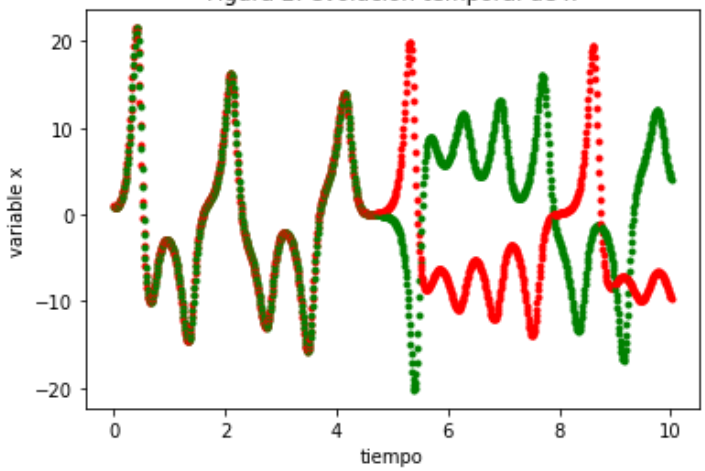


Figura 3: evolución temporal de y

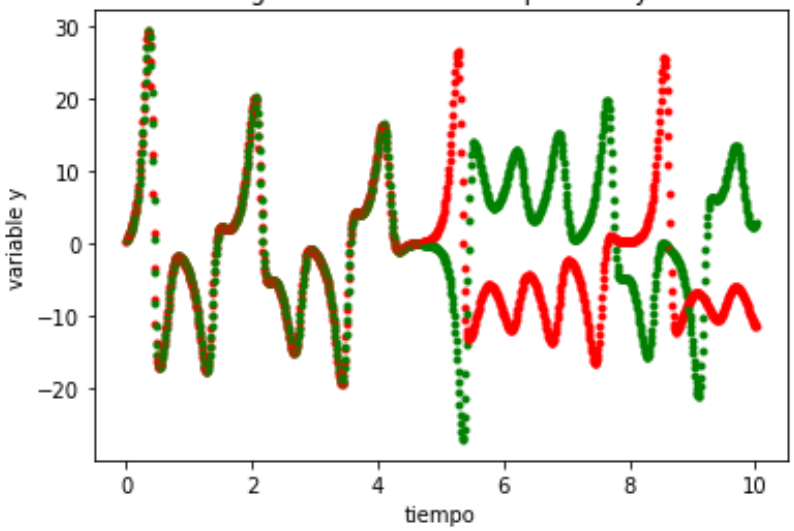
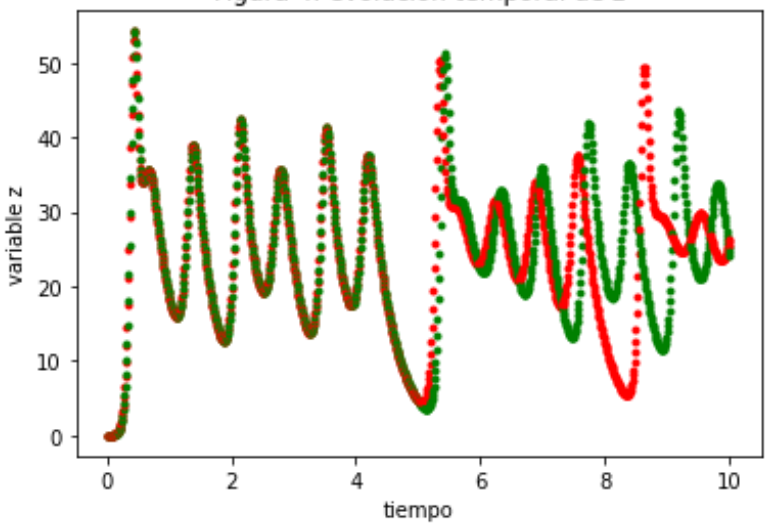
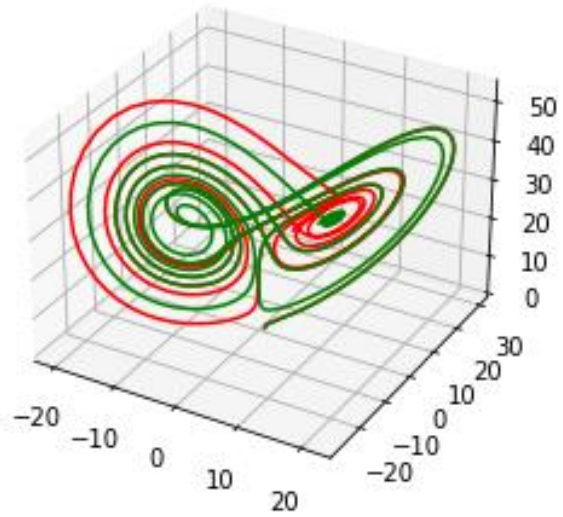


Figura 4: evolución temporal de z



Segunda parte: Gráficas a partir del código con vectores, por el método de Euler.

Figura 5: Atractor de Lorenz



A diferencia del gráfico de la figura 1, la distancia entre las dos curvas se visualiza mejor. De igual manera sucede con las gráficas de las evoluciones temporales ya que, en las realizadas a partir de funciones, los puntos no nos permitían distinguir con gran precisión las posiciones diferenciadas que tenía cada variable en diferentes instantes. Esto mejora al utilizar el método con vectores que traza la curva con una línea continua.

Figura 6: evolución temporal de x

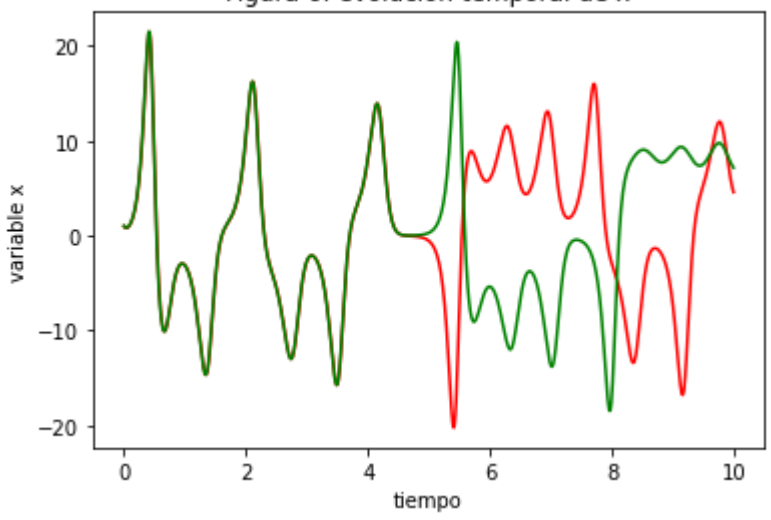


Figura 7: evolución temporal de y

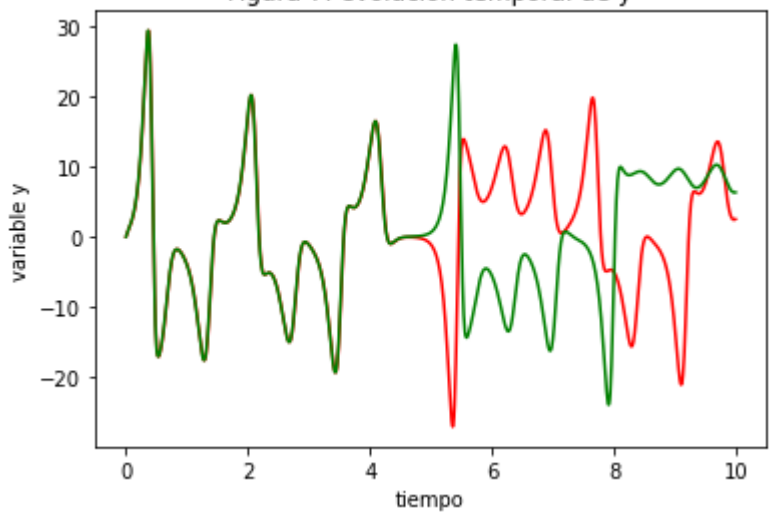
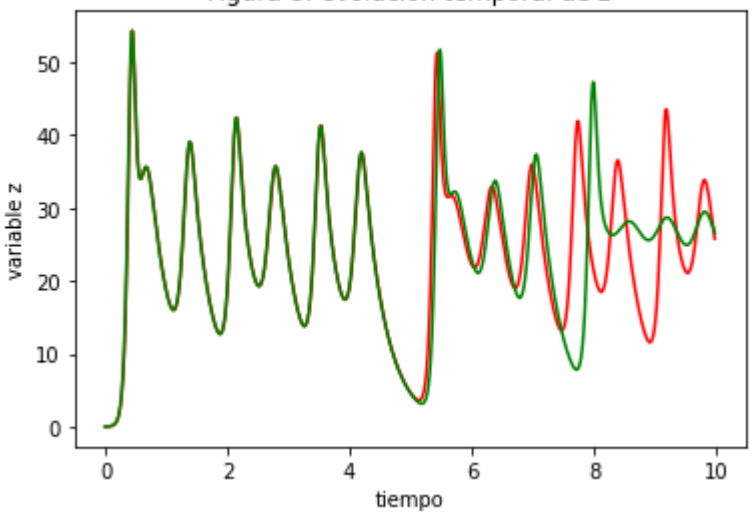


Figura 8: evolución temporal de z



A continuación, se presentan las gráficas de los cortes para los tres planos.

Figura 9: sección xy del atractor de Lorenz

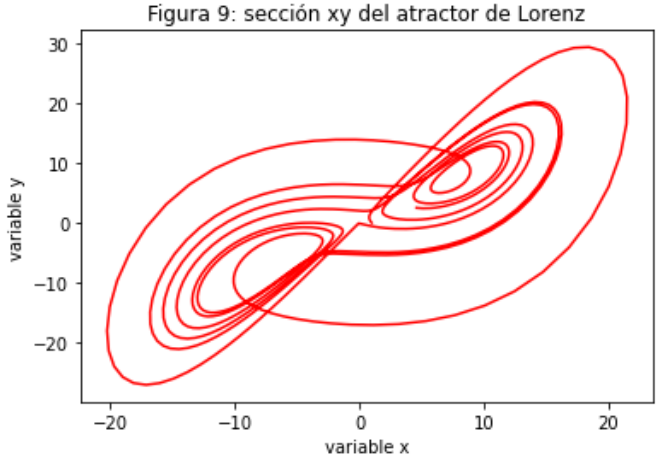


Figura 10: sección xz del atractor de Lorenz

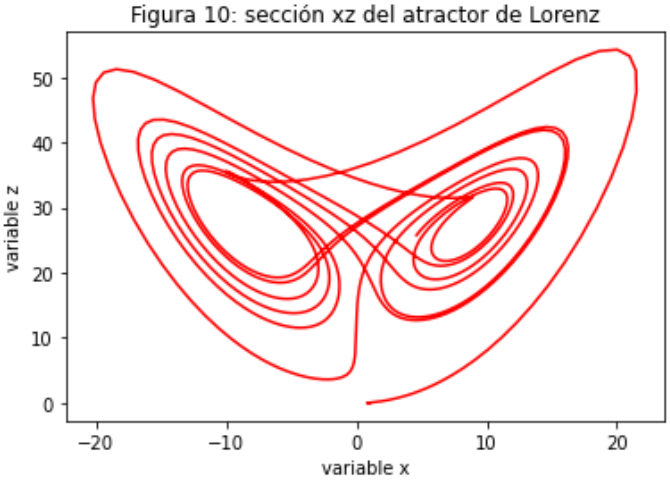
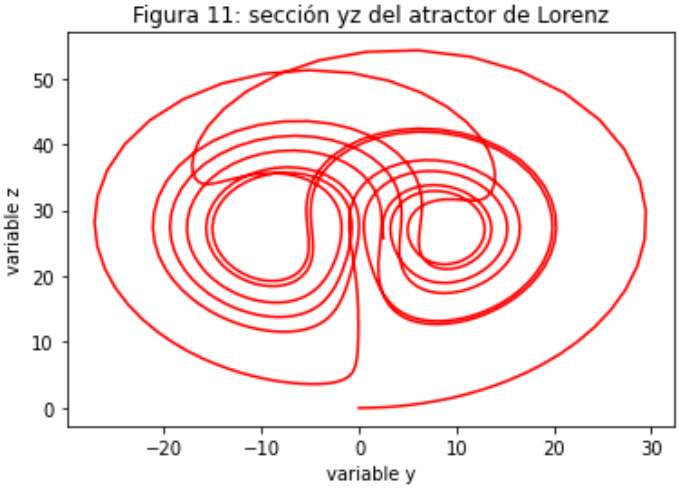


Figura 11: sección yz del atractor de Lorenz



Atractor de Rössler

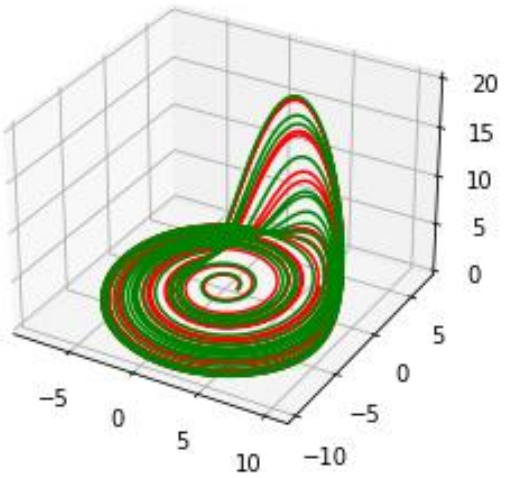
- ***Gráficas a partir del código con vectores, por el método de Euler:***

Se dividen en tres secciones:

1) **Gráficos con los parámetros a=b=0,2 y c=5**.

El gráfico 3d (figura 12) corresponde a los sistemas generados por diferentes condiciones iniciales. La roja representa al sistema determinado por (x,y,z)=(1;0;0) y la azul a (x2,y2,z2)=(1,1;0;0).

Figura 12: Atractor de Rösler



En la gráfica podemos distinguir que el sistema oscila alrededor del eje de coordenadas en el plano xy realizando saltos irregulares aumentando el valor de z.

Al igual que lo ocurrido con el anterior atractor, los sistemas a través del paso del tiempo divergen entre sí.

Figura 13: sección xy del atractor de Rösler

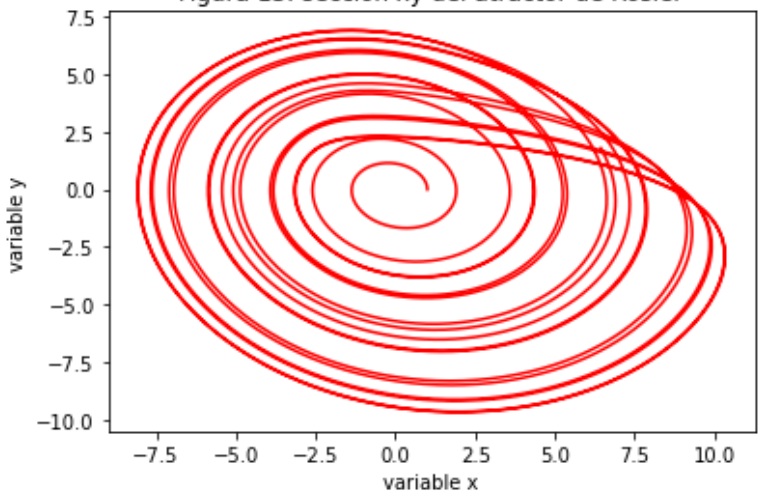


Figura 14: sección xz del atractor de Rösler

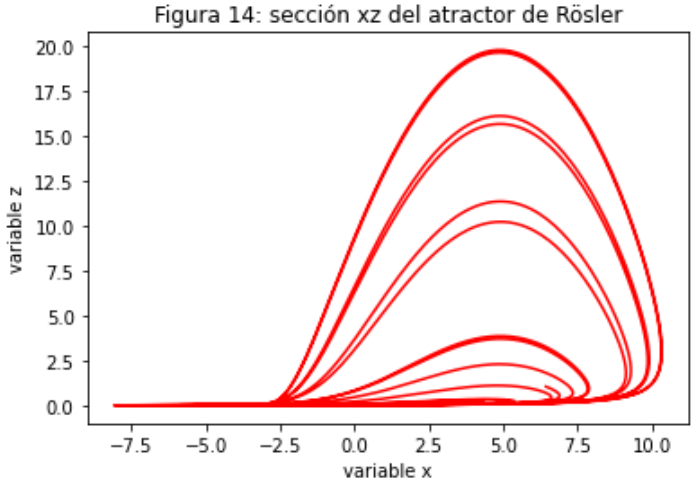
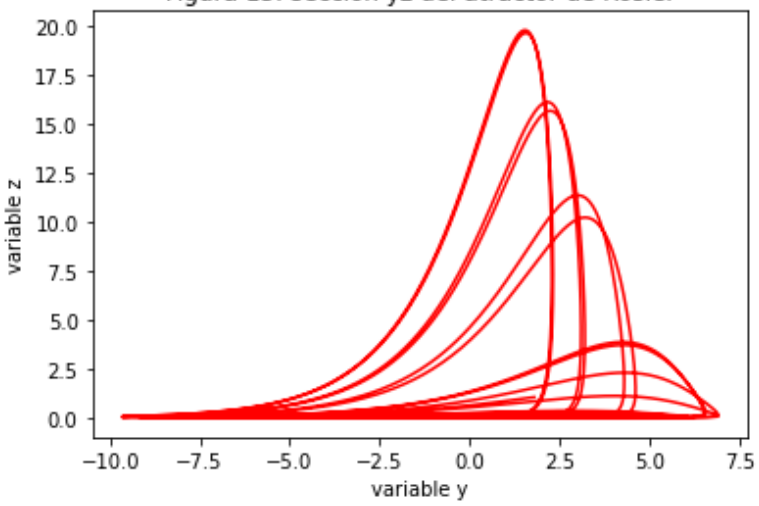


Figura 15: sección yz del atractor de Rösler



Por último, se presentan las gráficas de las evoluciones temporales.

Figura 16: evolución temporal de x

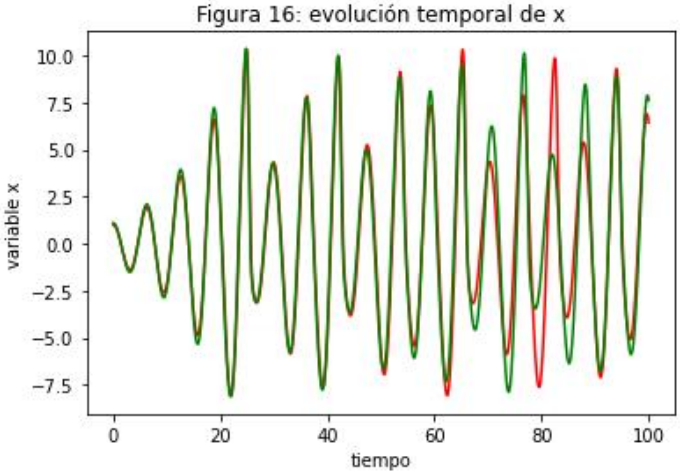


Figura 17: evolución temporal de y

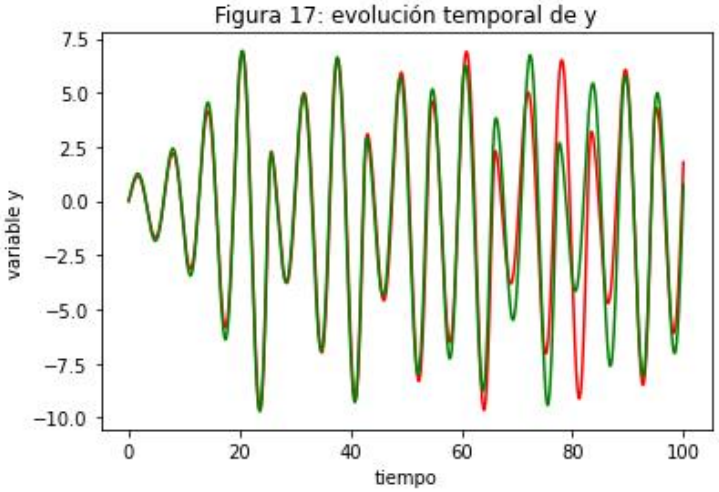
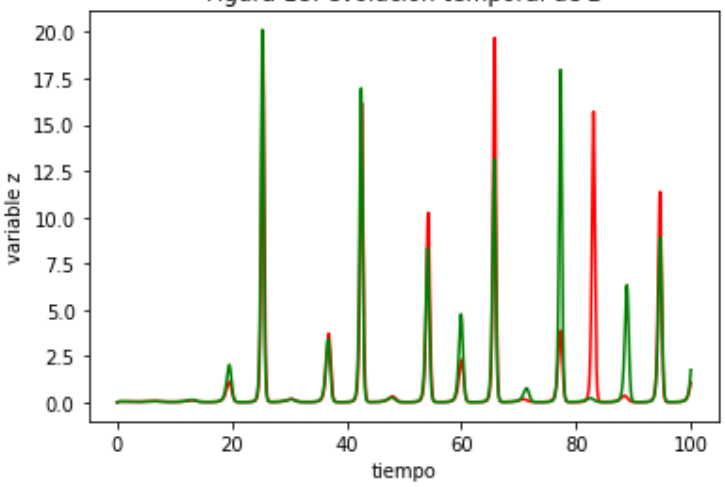


Figura 18: evolución temporal de z



2) Gráficos de las soluciones al variar el parámetro c.

Se designaron cuatro valores diferentes para el parámetro: 2.5, 3.5, 4 y 5, dejando que el sistema evolucionara los primeros 100 segundos hacia el atractor antes de comenzar a registrar los datos. A continuación, se representan sus gráficas respectivamente.

Figura 19: c = 2,5

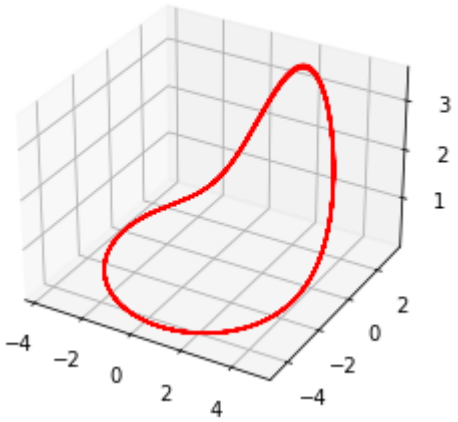


Figura 22: sección yz del atractor de Lorenz para c=3.5

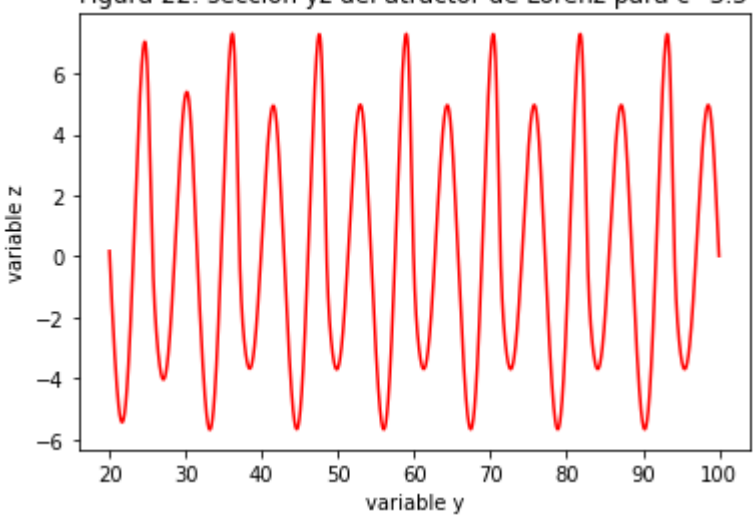


Figura 21: c = 3,5

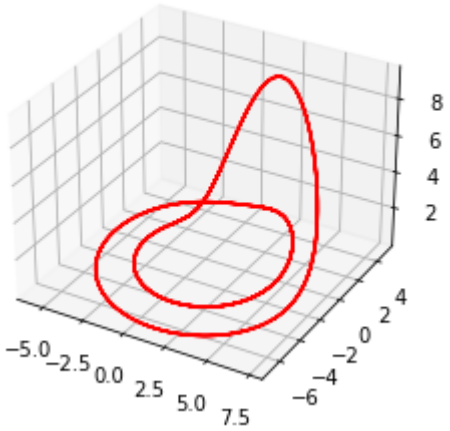


Figura 22: sección yz del atractor de Lorenz para c=3.5

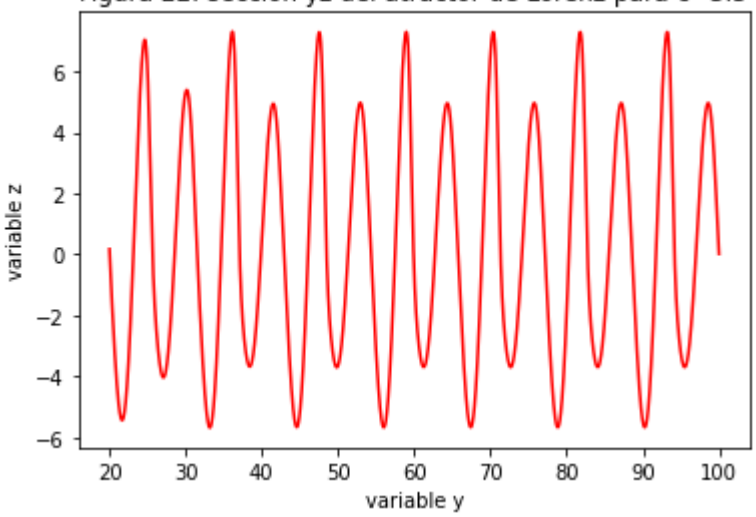


Figura 23: c = 4

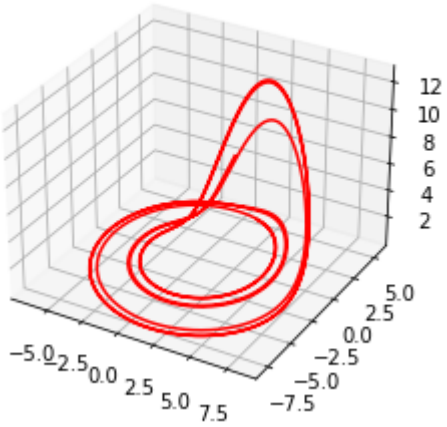


Figura 24: sección yz del atractor de Lorenz para c=4

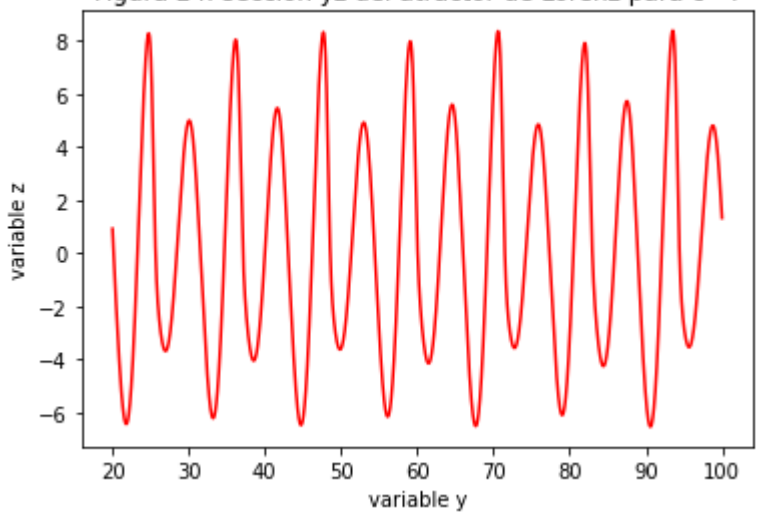
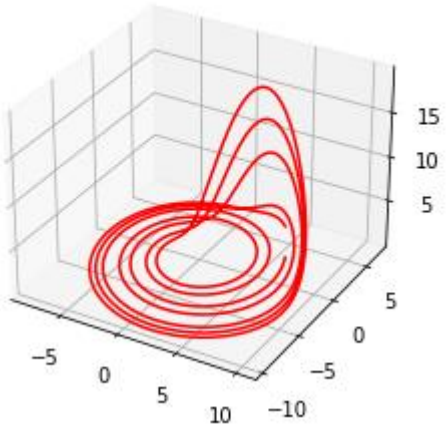
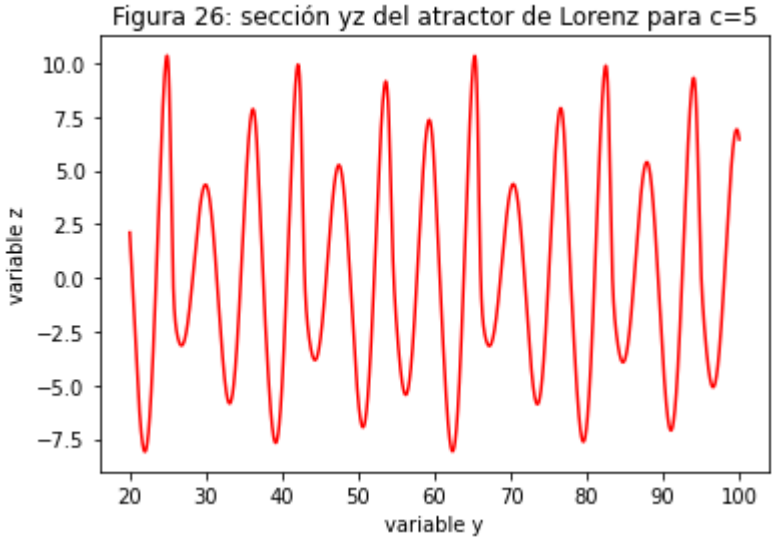


Figura 25: c = 5





Las figuras 19, 21 y 23 muestran órbitas de períodos 1, 2 y 4 respectivamente. Por otro lado, en la figura 25 se aprecia un comportamiento caótico.

3) Diagramas de bifurcación:

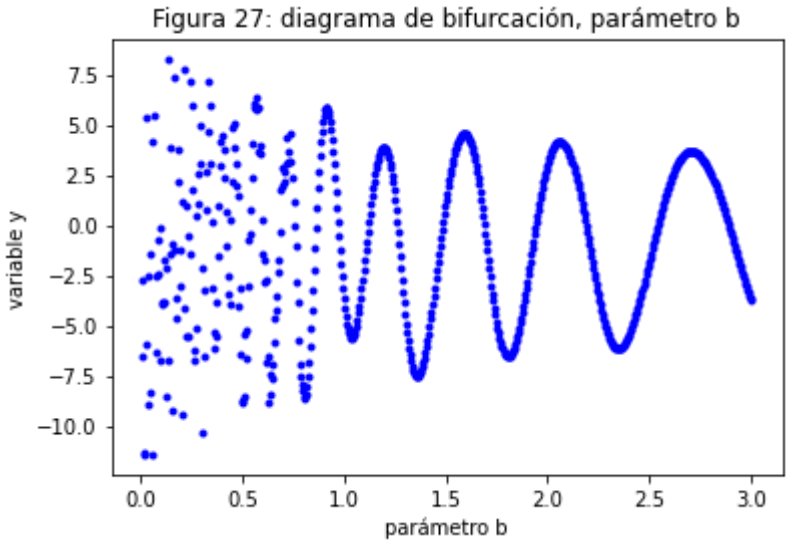
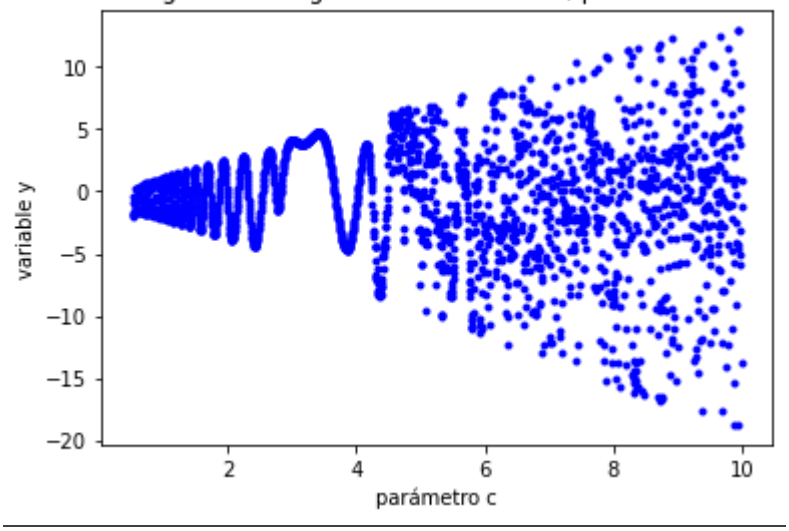


Figura 28: diagrama de bifurcación, parámetro c



Se puede apreciar que para valores de b pequeños (entre 0 y 1, aproximadamente), el sistema se comporta de forma caótica (lo visualizamos a partir de la nube de puntos presentes en este intervalo de tiempo). A medida que b crece, se tiende a perder el caos. Pero, al variar el parámetro c, ocurre lo contrario, ya que el comportamiento caótico aparece a partir de cierto valor (aproximadamente c=4) y se va incrementando (como lo anticipamos en las gráficas del atractor en tres dimensiones, al variar c).

ANÁLISIS Y CONCLUSIONES

Después de haber representado ambos atractores, podemos apreciar la característica principal de los sistemas caóticos: ante un pequeño cambio en las condiciones iniciales, las gráficas divergen rápidamente entre sí después de un cierto tiempo, llegando a ser completamente diferentes. Esto provoca que se vuelva imposible realizar predicciones a largo plazo, por más que las ecuaciones sean determinísticas.

Los recursos tecnológicos (en este caso, el lenguaje Python, utilizado con un mínimo conocimiento de programación básica) son de gran utilidad para estudiar los sistemas caóticos sin tener que resolver rebuscadas ecuaciones diferenciales, aún en los casos en que

no se sabe cómo resolverlas. De esta manera, conseguimos la solución con rapidez, permitiéndonos variar distintos parámetros para un mejor estudio de los problemas.

Por otro lado, para una mejor visualización y comprensión de las diferencias ocasionadas por las distintas condiciones iniciales en estos tipos de problemas, se recomienda utilizar el código empleando vectores para obtener una mejoría de los resultados al graficar.

El método de Euler, escogido para integrar numéricamente los sistemas, tiene por fundamento la aproximación de una curva mediante una secuencia de líneas rectas, lo que determina errores de cálculo desde un principio. A medida que nos alejamos del valor inicial, la solución aproximada pierde precisión (se aleja de la solución exacta). Para lograr mejores resultados, se puede disminuir el incremento, aunque esto provoca que, debido a la cantidad de iteraciones, el programa tarde mucho más tiempo en correr. Sin embargo, para el análisis cualitativo realizado en este trabajo, el método implica un código sencillo y se logra distinguir el comportamiento caótico, cumpliendo con el objetivo principal.

BIBLIOGRAFÍA CONSULTADA

- Boari, S. (2011). *Simulaciones numéricas de atractores extraños*. Departamento de Física. UBA.

- Bruzco, M.L. (2012). *El efecto mariposa y sus implicaciones estratégicas en el contexto organizacional.* Revista Ciencias Estratégicas, Vol. 20 (27), pp.39-49.

- De Régules, S. (2019). *Caos y complejidad: La realidad como un caleidoscopio.* Bonalletra Alcompras, S.L.

APÉNDICE

A continuación, se presentan los códigos utilizados en las simulaciones.

Atractor de Lorenz

❖ Código utilizando funciones con el método de Euler.

```
import matplotlib.pyplot as plt
def L(x,y,z):
    s=10
    r=28
    b=8/3
    dx=s*(y-x)
    dy=r*x-y-x*z
    dz=x*y-b*z
    return dx,dy,dz
t=0
dt=0.01
x,y,z=1,0,0
x2,y2,z2=1.1,0,0
fig1= plt.figure()
ax1 = plt.axes(projection='3d')
while t<10:
    dx,dy,dz=L(x,y,z)
    dx2,dy2,dz2=L(x2,y2,z2)
    x=dx*dt+x
    y=dy*dt+y
    z=dz*dt+z
    x2=dx2*dt+x2
    y2=dy2*dt+y2
    z2=dz2*dt+z2
    t=dt+t
    plt.figure(1)
    plt.title("Figura 1: Atractor de Lorenz")
    ax1.plot(x,y,z,".r")
    ax1.plot(x2,y2,z2,".b")
```

```
plt.figure(2)
plt.title("Figura 2: evolución temporal de x")
plt.xlabel('tiempo')
plt.ylabel('variable x')
plt.plot(t,x,".g")
plt.plot(t,x2,".r")
plt.figure(3)
plt.title("Figura 3: evolución temporal de y")
plt.xlabel('tiempo')
plt.ylabel('variable y')
plt.plot(t,y,".g")
plt.plot(t,y2,".r")
plt.figure(4)
plt.title("Figura 4: evolución temporal de z")
plt.xlabel('tiempo')
plt.ylabel('variable z')
plt.plot(t,z,".g")
plt.plot(t,z2,".r")
```

- **Código utilizando vectores con el método de Euler.**

```
import math
import numpy as np
import matplotlib.pyplot as plt

#Atractor de Lorenz

#Tres variables: x, y, z
#Tres parámetros: sigma (s), ro (r) y beta (b)

s=10
r=28
```

```
b=8/3

N=1000

x=np.zeros(N)
x[0]=1
y=np.zeros(N)
y[0]=0
z=np.zeros(N)
z[0]=0

x2=np.zeros(N)
x2[0]=1.1
y2=np.zeros(N)
y2[0]=0
z2=np.zeros(N)
z2[0]=0

dx=np.zeros(N)
dx[0]=s*(y[0]-x[0])
dy=np.zeros(N)
dy[0]=x[0]*(r-z[0])-y[0]
dz=np.zeros(N)
dz[0]=x[0]*y[0]-b*z[0]

dx2=np.zeros(N)
dx2[0]=s*(y2[0]-x2[0])
dy2=np.zeros(N)
dy2[0]=x2[0]*(r-z2[0])-y2[0]
dz2=np.zeros(N)
dz2[0]=x2[0]*y2[0]-b*z2[0]

t=np.zeros(N)
```

```
t[0]=0
dt=0.01

ax1 = plt.axes(projection='3d')

i=0

while i<np.size(t)-1: #hasta que se llene el vector
  x[i+1]=x[i]+dx[i]*dt
  y[i+1]=y[i]+dy[i]*dt
  z[i+1]=z[i]+dz[i]*dt

  x2[i+1]=x2[i]+dx2[i]*dt
  y2[i+1]=y2[i]+dy2[i]*dt
  z2[i+1]=z2[i]+dz2[i]*dt

  dx[i+1]=s*(y[i+1]-x[i+1])
  dy[i+1]=x[i+1]*(r-z[i+1])-y[i+1]
  dz[i+1]=x[i+1]*y[i+1]-b*z[i+1]

  dx2[i+1]=s*(y2[i+1]-x2[i+1])
  dy2[i+1]=x2[i+1]*(r-z2[i+1])-y2[i+1]
  dz2[i+1]=x2[i+1]*y2[i+1]-b*z2[i+1]

  t[i+1]=t[i]+dt

  i=i+1

plt.figure(1)
plt.title("Figura 5: Atractor de Lorenz")
ax1.plot(x[:i],y[:i],z[:i],"r")
ax1.plot(x2[:i],y2[:i],z2[:i],"g")
```

```
plt.figure(2)
plt.title("Figura 6: evolución temporal de x")
plt.xlabel('tiempo')
plt.ylabel('variable x')
plt.plot(t[:i],x[:i],"r")
plt.plot(t[:i],x2[:i],"g")

plt.figure(3)
plt.title("Figura 7: evolución temporal de y")
plt.xlabel('tiempo')
plt.ylabel('variable y')
plt.plot(t[:i],y[:i],"r")
plt.plot(t[:i],y2[:i],"g")

plt.figure(4)
plt.title("Figura 8: evolución temporal de z")
plt.xlabel('tiempo')
plt.ylabel('variable z')
plt.plot(t[:i],z[:i],"r")
plt.plot(t[:i],z2[:i],"g")

plt.figure(5)
plt.title("Figura 9: sección xy del atractor de Lorenz")
plt.xlabel('variable x')
plt.ylabel('variable y')
plt.plot(x[:i],y[:i],"r")

plt.figure(6)
plt.title("Figura 10: sección xz del atractor de Lorenz")
plt.xlabel('variable x')
plt.ylabel('variable z')
plt.plot(x[:i],z[:i],"r")
```

```
plt.figure(7)
plt.title("Figura 11: sección yz del atractor de Lorenz")
plt.xlabel('variable y')
plt.ylabel('variable z')
plt.plot(y[:i],z[:i],"r")
```

Atractor de Rossler

❖ Código con vectores utilizando el método de Euler.

```
import math
import numpy as np
import matplotlib.pyplot as plt

#Atractor de Rossler

#Tres variables: x, y, z
#Tres parámetros: a, b, c

a=0.2
b=0.2
c=5

N=10000

x=np.zeros(N)
x[0]=1
y=np.zeros(N)
y[0]=0
z=np.zeros(N)
z[0]=0
```

```
x2=np.zeros(N)
x2[0]=1.1
y2=np.zeros(N)
y2[0]=0
z2=np.zeros(N)
z2[0]=0

dx=np.zeros(N)
dx[0]=-y[0]-z[0]
dy=np.zeros(N)
dy[0]=x[0]+(a*y[0])
dz=np.zeros(N)
dz[0]=b+(z[0]*(x[0]-c))

dx2=np.zeros(N)
dx2[0]=-y2[0]-z2[0]
dy2=np.zeros(N)
dy2[0]=x2[0]+(a*y2[0])
dz2=np.zeros(N)
dz2[0]=b+(z2[0]*(x2[0]-c))

t=np.zeros(N)
t[0]=0
dt=0.01

ax1 = plt.axes(projection='3d')

i=0

while i<np.size(t)-1: #hasta que se llene el vector
  x[i+1]=x[i]+dx[i]*dt
  y[i+1]=y[i]+dy[i]*dt
  z[i+1]=z[i]+dz[i]*dt
```

```
x2[i+1]=x2[i]+dx2[i]*dt
y2[i+1]=y2[i]+dy2[i]*dt
z2[i+1]=z2[i]+dz2[i]*dt

dx[i+1]=-y[i+1]-z[i+1]
dy[i+1]=x[i+1]+(a*y[i+1])
dz[i+1]=b+(z[i+1]*(x[i+1]-c))

dx2[i+1]=-y2[i+1]-z2[i+1]
dy2[i+1]=x2[i+1]+(a*y2[i+1])
dz2[i+1]=b+(z2[i+1]*(x2[i+1]-c))

t[i+1]=t[i]+dt

i=i+1

plt.figure(1)
plt.title("Figura 12: Atractor de Rösler")
ax1.plot(x[:i],y[:i],z[:i],"r")
ax1.plot(x2[:i],y2[:i],z2[:i],"g")

plt.figure(2)
plt.title("Figura 16: evolución temporal de x")
plt.xlabel('tiempo')
plt.ylabel('variable x')
plt.plot(t[:i],x[:i],"r")
plt.plot(t[:i],x2[:i],"g")

plt.figure(3)
plt.title("Figura 17: evolución temporal de y")
plt.xlabel('tiempo')
plt.ylabel('variable y')
```

```
plt.plot(t[:i],y[:i],"r")
plt.plot(t[:i],y2[:i],"g")

plt.figure(4)
plt.title("Figura 18: evolución temporal de z")
plt.xlabel('tiempo')
plt.ylabel('variable z')
plt.plot(t[:i],z[:i],"r")
plt.plot(t[:i],z2[:i],"g")

plt.figure(5)
plt.title("Figura 13: sección xy del atractor de Rösler")
plt.xlabel('variable x')
plt.ylabel('variable y')
plt.plot(x[:i],y[:i],"r")

plt.figure(6)
plt.title("Figura 14: sección xz del atractor de Rösler")
plt.xlabel('variable x')
plt.ylabel('variable z')
plt.plot(x[:i],z[:i],"r")

plt.figure(7)
plt.title("Figura 15: sección yz del atractor de Rösler")
plt.xlabel('variable y')
plt.ylabel('variable z')
plt.plot(y[:i],z[:i],"r")
```

- **Código utilizado para graficar a partir de los 100 segundos**

```
#Atractor de Rössler VARIANDO C

#Tres variables: x, y, z
```

```
#Tres parámetros: a, b, c

a=0.2
b=0.2
c=5

N=10000

x=np.zeros(N)
x[0]=1
y=np.zeros(N)
y[0]=0
z=np.zeros(N)
z[0]=0

dx=np.zeros(N)
dx[0]=-y[0]-z[0]
dy=np.zeros(N)
dy[0]=x[0]+(a*y[0])
dz=np.zeros(N)
dz[0]=b+(z[0]*(x[0]-c))

t=np.zeros(N)
t[0]=0
dt=0.01

ax1 = plt.axes(projection='3d')

i=0

while i<np.size(t)-1: #hasta que se llene el vector
  x[i+1]=x[i]+dx[i]*dt
  y[i+1]=y[i]+dy[i]*dt
```

```
z[i+1]=z[i]+dz[i]*dt

dx[i+1]=-y[i+1]-z[i+1]
dy[i+1]=x[i+1]+(a*y[i+1])
dz[i+1]=b+(z[i+1]*(x[i+1]-c))

t[i+1]=t[i]+dt

i=i+1

plt.figure(1)
plt.title("Figura 25: c = 5")
ax1.plot(x[6000:i],y[6000:i],z[6000:i],"r")

plt.figure(2)
plt.title("Figura 26: sección yz del atractor de Lorenz para c=5")
plt.xlabel('variable y')
plt.ylabel('variable z')
plt.plot(t[2000:i],x[2000:i],"r")
```

❖ Diagrama de bifurcación variando b

```
import matplotlib.pyplot as plt
def L(x,y,z,b):
    a=0.2
    c=5.7
    dx=-y-z
    dy=x+(a*y)
    dz=b+(z*(x-c))
    return dx,dy,dz


db=0.005
```

```
b=0

while b<3:
    b=b+db

    t=0
    dt=0.01
    x,y,z=0,1,0

    while t<1000:
        dx,dy,dz=L(x,y,z,b)
        x=dx*dt+x
        y=dy*dt+y
        z=dz*dt+z
        t=dt+t
    plt.title("Figura 27: diagrama de bifurcación, parámetro b")
    plt.xlabel('parámetro b')
    plt.ylabel('variable y')
    plt.plot(b,y,".b")
```

- **Diagrama de bifurcación variando c**

```
import matplotlib.pyplot as plt
def L(x,y,z,b):
    a=0.2
    b=0.2
    dx=-y-z
    dy=x+(a*y)
    dz=b+(z*(x-c))
    return dx,dy,dz


dc=0.005
```

```
c=0

while c<10:
    c=c+dc

    t=0
    dt=0.01
    x,y,z=0,1,0

    while t<1000:
        dx,dy,dz=L(x,y,z,c)
        x=dx*dt+x
        y=dy*dt+y
        z=dz*dt+z
        t=dt+t
    plt.title("Figura 28: diagrama de bifurcación, parámetro c")
    plt.xlabel('parámetro c')
    plt.ylabel('variable y')
    plt.plot(c,y,".b")
```