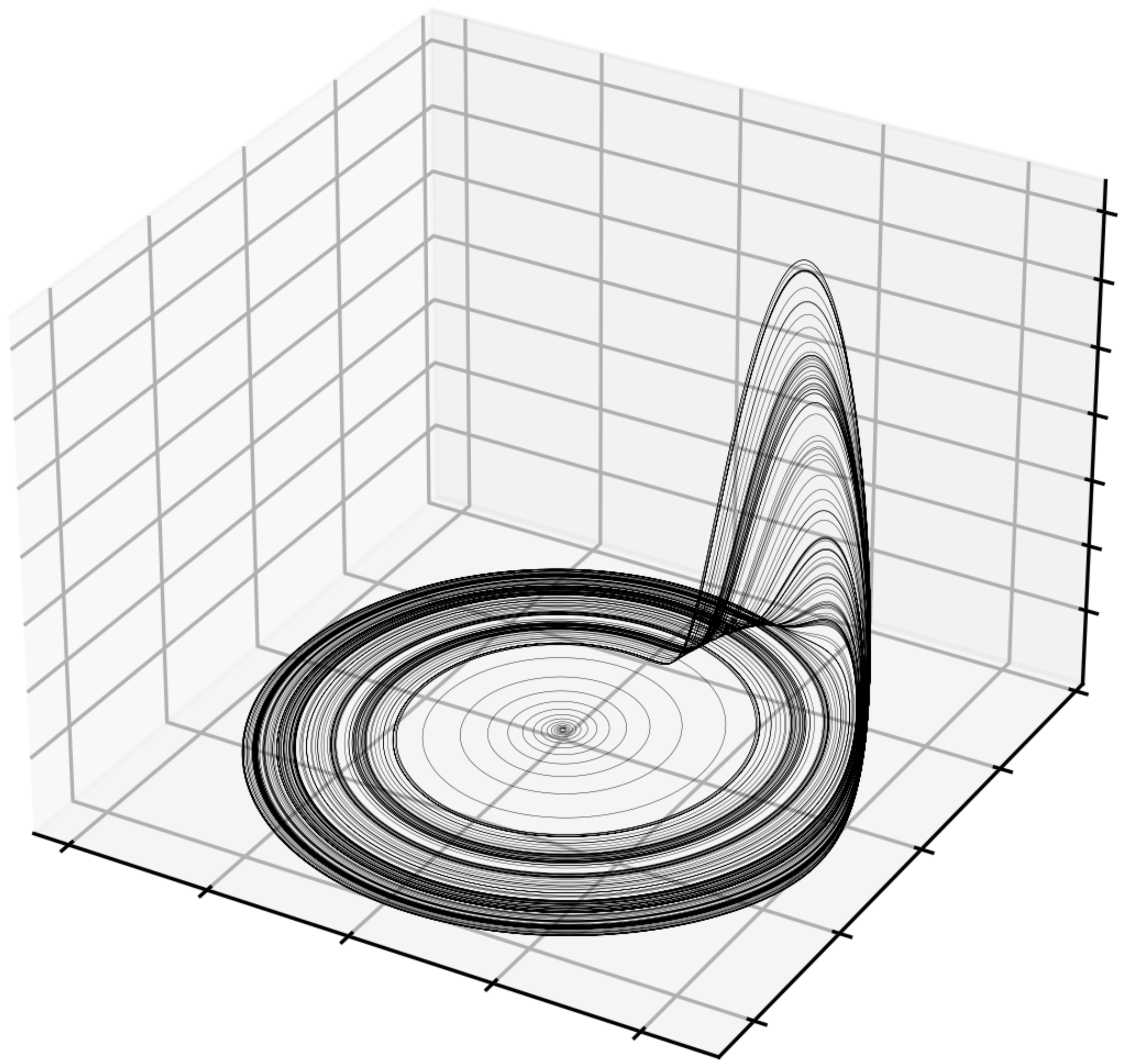
Term Paper

The Rossler Attractor



Abstract

The following term paper showcases my understanding of a topic of my choice after having taken this course on Nonlinear Dynamics. The Rossler attractor being closely related to the Lorenz attractor but not quite as known as it, was chosen as the topic of this term paper.

Introduction

Proposed by Dr. Otto E. Rossler (1976) , the solution of the following set of ordinary differential equations is known as The Rossler attractor. Exhibiting chaotic dynamics (as shown in the figure above for a = 0.1, b = 0.1 & c = 14) such as the Lorenz attractor, this particular system has only one sink and only one nonlinear term. The Lyapunov exponents are approximately 0.072 and -13.79 with the corresponding Lyapunov dimension being 2.005.

$$\begin{aligned}\dot{x} &= -y - z\\ \dot{y} &= x + ay\\ \dot{z} &= b + z(x - c)\end{aligned}$$

The behaviour of this attractor can be intuitively understood by "switching off" z coordinate observing only the x and y terms.

$$\begin{aligned}\dot{x} &= -y\\ \dot{y} &= x + ay\end{aligned}$$

At the origin $(0, 0)$ both $\dot{x} = 0$ and $\dot{y} = 0$. At $(0, 0)$, the jacobian is given as

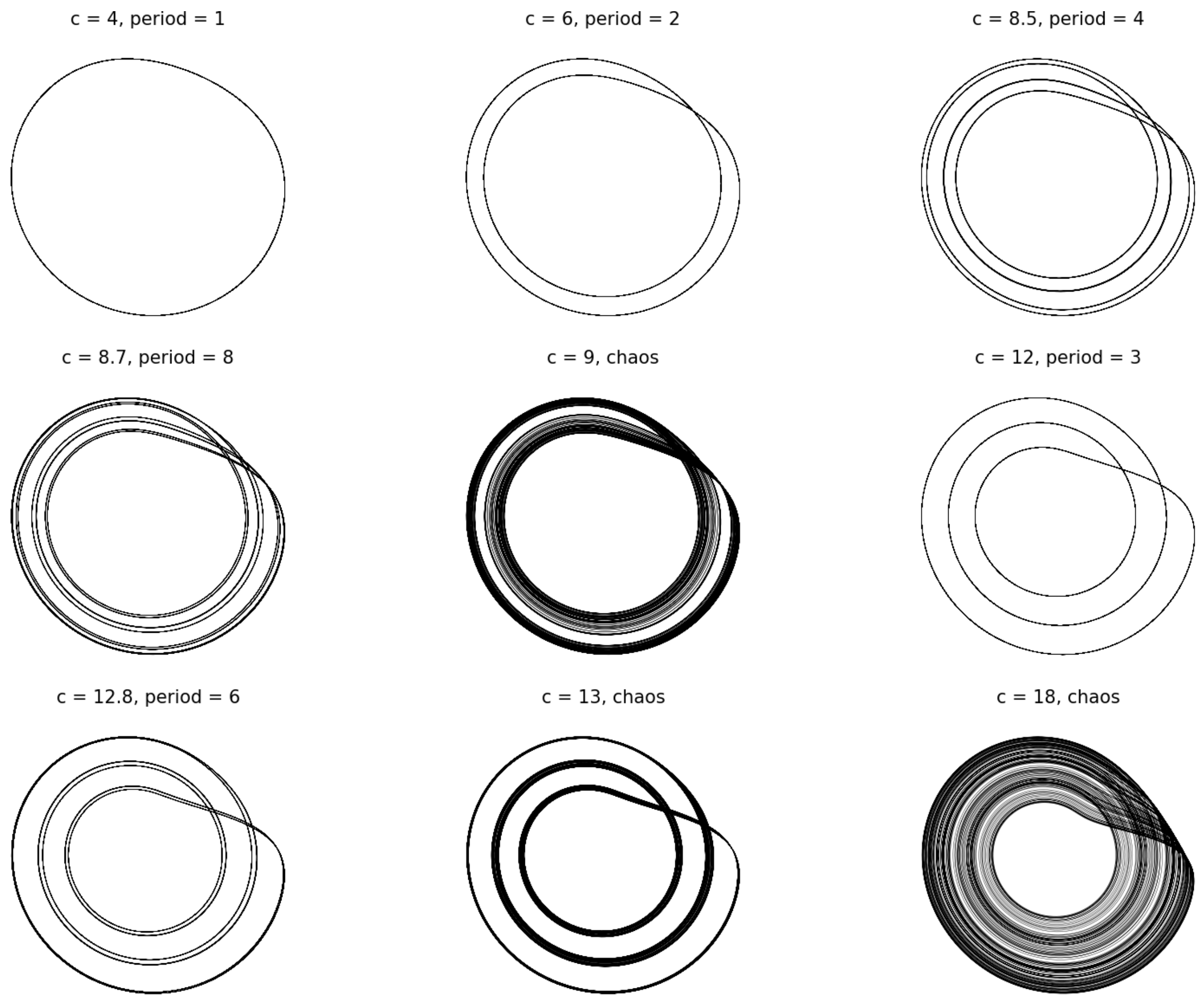
$$J = \begin{pmatrix} 0 & -1 \\ 1 & a \end{pmatrix}$$

The eigenvalues for this jacobian are given as $(a \pm \sqrt{a^2 - 4})/2$. For $a > 0$, atleast one eigenvalue is positive making the origin an unstable point. For $0 < a < 2$ eigenvalues are complex, hinting that the trajectories spiral out near the origin in the xy plane when x near zero.

Now assume $0 < a < 2$ and $b, c > 0$. We start with $z \approx 0$ and near the origin. The trajectory spirals out in the xy plane as long as the x coordinate is smaller than c. When x grows greater than c, the z values grow since $\dot{z} > 0$. Larger z values decrease $\dot{x}$ as visible in the first equation of the rossler attractor.

This way the x and z variables influence each other and keep the system bounded. Trajectories spiral out from the xy plane, get extruded in the z direction and eventually get reinserted near the xy plane. The attractor topologically appears to be a combination of two surfaces, a circular disk and a möbius strip.

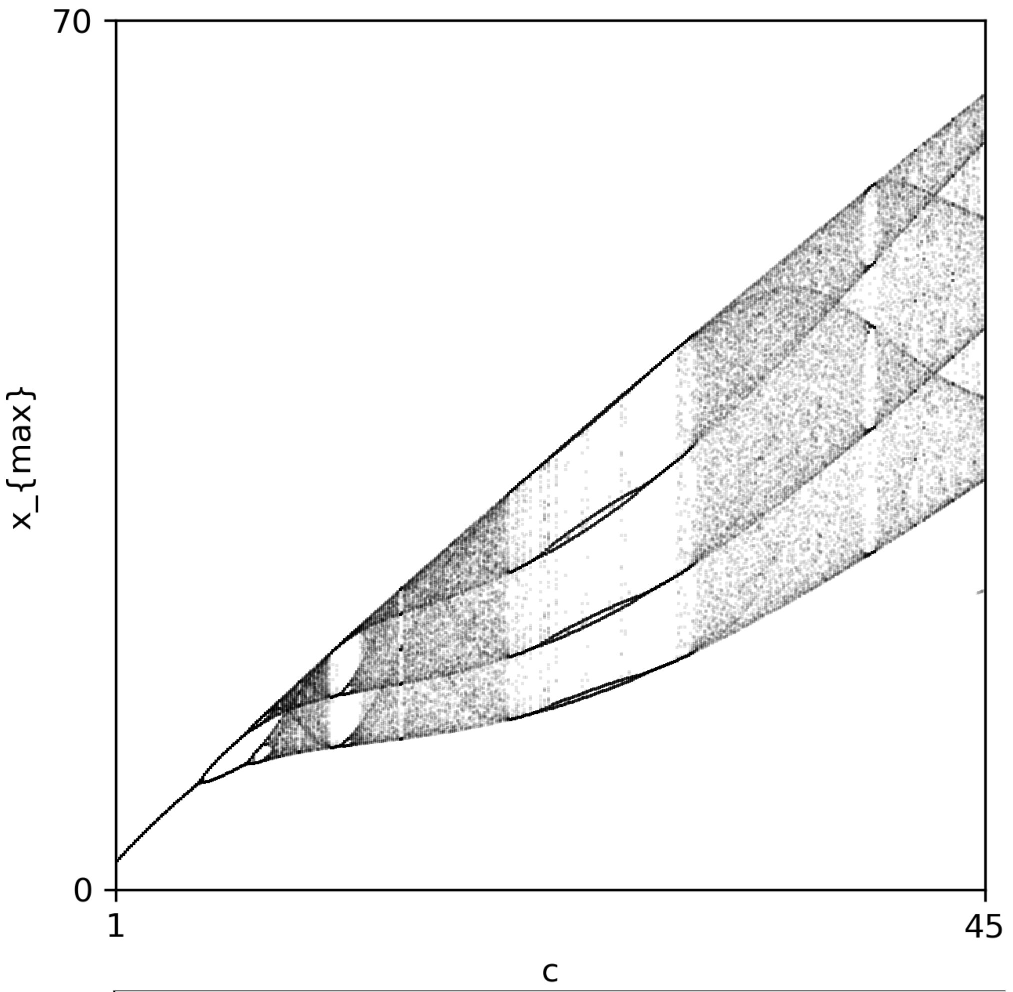
Changing parameters

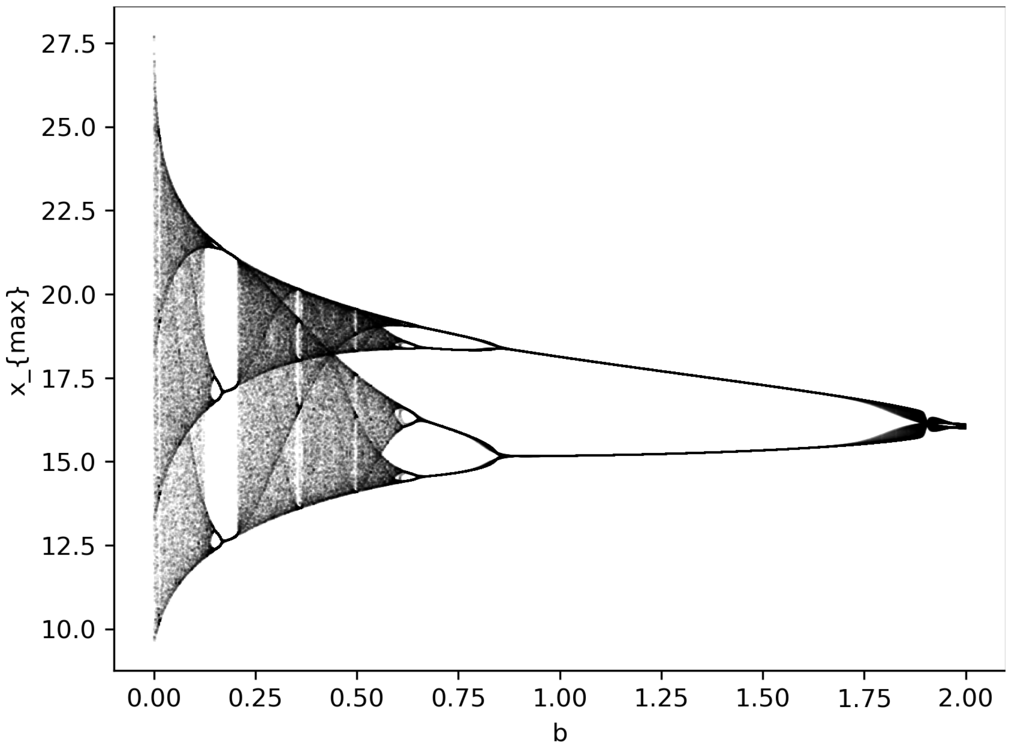


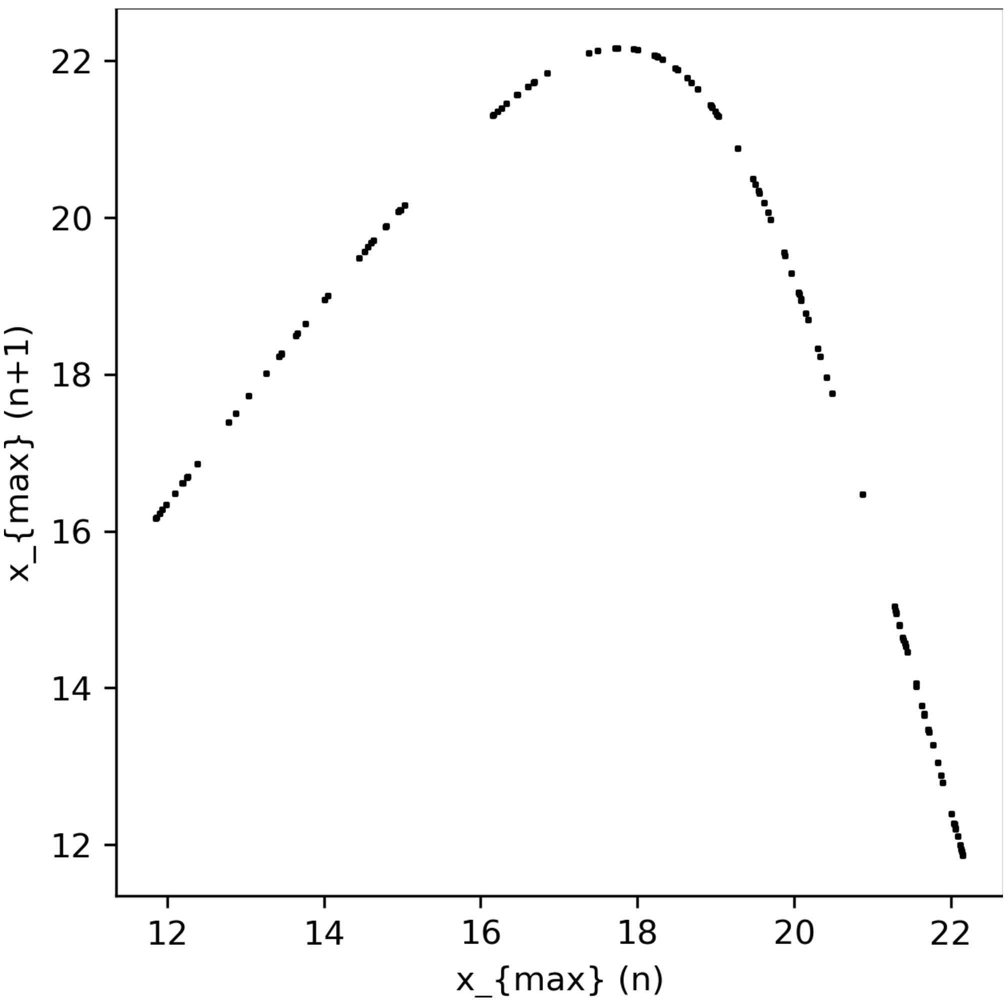
Given above are the xy plane projections of the rossler attractor for different values of the parameter c while a, b are held constant at 0.1.

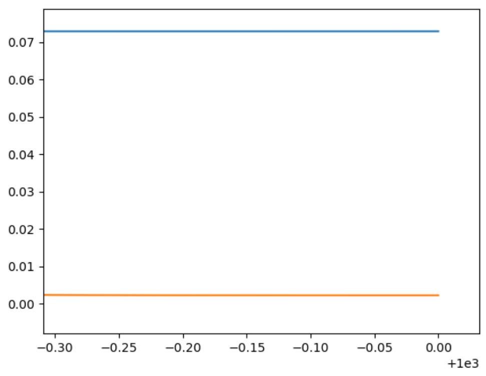
Given below are bifurcation diagrams for the parameters b and c. For the first bifurcation diagram we fix the parameters a and b at 0.1 and sweep through a range from 1 to 45 of parameter c. For the second we fix the parameters b and c at 0.1 and 14 respectively and sweep through a range from 0 to 2 of parameter b. After obtaining the parameters we plot the local maxima values for x coordinate to obtain these figures.

We can see structures similar to the iconic logistic model. The pitchfork bifurcation in the start, period doubling cascade and the period 3 window. Similar structures can be obtained for parameter a as well. Also given below is a map using Lorenz's trick for the rossler attractor by plotting x_{n+1} vs x_n where x_n denotes nth local maxima which is quite similar to the logistic map.

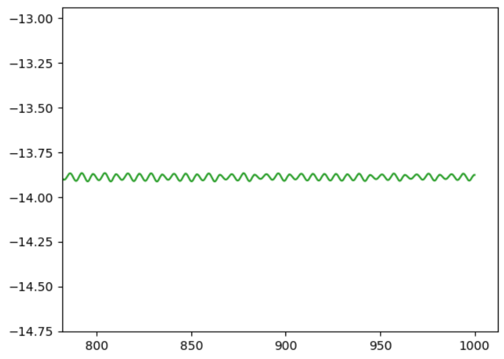

70
0
1
45
c
x_{max}


27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
b
x_{max}





(a) First two of the lyapunov exponents converging to 0 and 0.072 respectively



(b) Third lyapunov exponent converging to -13.79

This concludes this term paper. The code and refrences are mentioned in the later sections

Code for generating figures

```
#code for idc402 nonlinear term paper
#may take 10 min to generate all plots
#preferrably use python notebook to run
#different snippets of code to get plots
#generated faster
#importing required modules
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.integrate import odeint
from scipy.signal import find_peaks
#change frontend for jupyter notebooks
#%matplotlib qt
#progression rule
def rossler(state, t, a, b, c):
    x, y, z = state
    return -y - z, x + a*y, b + z*(x - c)
#plot main rossler
state0 = np.random.random(3)
t = np.arange(0.0, 1000.0, 0.01)

states = odeint(rossler, state0, t, args=(0.1,0.1,14))
fig = plt.figure(dpi = 300)
ax = plt.axes(projection='3d')
ax.plot(states[:,0], states[:,1], states[:,2], color = 'k', linewidth = 0.
                                              1)
ax.set_xticklabels([])
ax.set_zticklabels([])
ax.set_yticklabels([])
plt.savefig('rossler_main.png', dpi = 300, bbox_inches='tight',pad_inches
                                              = 0)
#plot attractor in 3x3 grid for certain c values
fig, ax = plt.subplots(3,3, dpi = 300)
states = odeint(rossler, state0, t, args=(0.1,0.1,4))
ax[0,0].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                              0.1)
ax[0,0].set_aspect(1)
ax[0,0].set_axis_off()
ax[0,0].set_title('c = 4, period = 1', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,6))
ax[0,1].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                              0.1)
ax[0,1].set_aspect(1)
ax[0,1].set_axis_off()
ax[0,1].set_title('c = 6, period = 2', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,8.5))
```

```
ax[0,2].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[0,2].set_aspect(1)
ax[0,2].set_axis_off()
ax[0,2].set_title('c = 8.5, period = 4', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,8.7))
ax[1,0].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[1,0].set_aspect(1)
ax[1,0].set_axis_off()
ax[1,0].set_title('c = 8.7, period = 8', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,9))
ax[1,1].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[1,1].set_aspect(1)
ax[1,1].set_axis_off()
ax[1,1].set_title('c = 9, chaos', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,12))
ax[1,2].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[1,2].set_aspect(1)
ax[1,2].set_axis_off()
ax[1,2].set_title('c = 12, period = 3', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,12.8))
ax[2,0].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[2,0].set_aspect(1)
ax[2,0].set_axis_off()
ax[2,0].set_title('c = 12.8, period = 6', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,13))
ax[2,1].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[2,1].set_aspect(1)
ax[2,1].set_axis_off()
ax[2,1].set_title('c = 13, chaos', size = 5)
states = odeint(rossler, state0, t, args=(0.1,0.1,18))
ax[2,2].plot(states[20000:,0], states[20000:,1], color = 'k', linewidth =
                                        0.1)
ax[2,2].set_aspect(1)
ax[2,2].set_axis_off()
ax[2,2].set_title('c = 18, chaos', size = 5)
plt.savefig('rossler_c.png', dpi = 300, bbox_inches='tight',pad_inches = 0
                                        )
#bifurcation diagram
#cvalues to sweep
c_val = np.arange(1,45,0.1)
fig, ax = plt.subplots()
for c in c_val:
    states = odeint(rossler, state0, t, args=(0.1,0.1,c))
    peaks, _ = find_peaks(states[20000:,0], height=0)
    ax.scatter(np.ones_like(peaks)*c, states[20000:,0][peaks], color = 'k'
                                            , marker = 's', s = 0.6,
                                            linewidths=0, alpha= 0.1)
ax.set_xlim(1,45)
ax.set_ylim(0,70)
ax.set_xlabel(r'c')
ax.set_ylabel(r'x_{max}')
ax.set_xticks([1,45])
ax.set_yticks([0,70])
ax.set_xticklabels([1,45])
ax.set_yticklabels([0,70])
ax.set_aspect(44/70)
plt.savefig('rossler_bifurcation_c.png', dpi = 300, bbox_inches='tight',
                                      pad_inches = 0)
#bifurcation diagram
#bvalues to sweep
b_val = np.arange(0,2,0.001)
fig, ax = plt.subplots()
for b in b_val:
    states = odeint(rossler, state0, t, args=(0.1,b,14))
    peaks, _ = find_peaks(states[20000:,0], height=0)
    ax.scatter(np.ones_like(peaks)*b, states[20000:,0][peaks], color = 'k'
                                          , marker = 's', s = 0.6,
                                          linewidths=0, alpha=0.1)
ax.set_xlabel(r'b')
ax.set_ylabel(r'x_{max}')
plt.savefig('rossler_bifurcation_b.png', dpi = 300, bbox_inches='tight',
                                      pad_inches = 0)
#lorenz map
states = odeint(rossler, state0, t, args = (0.1, 0.1, 14))
peaks, _ = find_peaks(states[20000:,0], height=0)
fig, ax = plt.subplots(dpi = 300)
ax.scatter(states[20000:,0][peaks][:-1], states[20000:,0][peaks][1:],
                                      color = 'k', s = 1, marker = 's')
ax.set_aspect(1)
ax.set_xlabel(r'x_{max} (n)')
ax.set_ylabel(r'x_{max} (n+1)')
plt.savefig('rossler_lorenz_map', dpi = 300, bbox_inches = 'tight',
                                      pad_inches = 0)
```

```
#Credits to Lutz Lehmann for this code
#originally for lorenz attractor
#modified for rossler
def diff_Lorenz(u):
    x,y,z = u
    f = [-y-z, x + 0.1*y, 0.1 + z*(x-14)]
    Df = [[0,-1,-1], [1,0.1,0], [z,0,x-14]]
    return np.array(f), np.array(Df)

def LEC_system(u):
    #x,y,z = u[:3]             # n=3
    U = u[3:12].reshape([3,3]) # size n square matrix, sub-array from n to
                                          n+n*n=n*(n+1)
    L = u[12:15]               # vector, sub-array from n*(n+1) to n*(n+1)
                                         +n=n*(n+2)
    f,Df = diff_Lorenz(u[:3])
    A = U.T.dot(Df.dot(U))
    dL = np.diag(A).copy();
    for i in range(3):
        A[i,i] = 0
        for j in range(i+1,3): A[i,j] = -A[j,i]
    dU = U.dot(A)
    return np.concatenate([f,dU.flatten(),dL])

u0 = np.ones(3)
U0 = np.identity(3)
L0 = np.zeros(3)
u0 = np.concatenate([u0, U0.flatten(), L0])
t = np.linspace(0,1000,100000)
u = odeint(lambda u,t:LEC_system(u),u0,t, hmax=0.05)
L = u[5:,12:15].T/t[5:]

plt.plot(t[5:],L.T)
```

Refrences

- NONLINEAR DYNAMICS AND CHAOS by
  Steveh H. Strogatz

- CHAOS an Introduction to dynamical systems by
  KATHlEEN T. AlllGOOD
  George Mason University
  TIM D. SAUER
  George Mason University
  JAMES A. YORKE

- Code for Numerical Computation of lyapunov exponent of continuos flows
  Click Here

— ∗ ∗ ∗ ∗ ∗ —