An Introductory Study of Strange Attractors through Some Relevant Systems

Jai Israni
Mithil Vakde
Priyam Dubey
Shyam Iyer

Abstract

Strange attractors in phase space are a recent development in the study of dynamical systems, noted for their unique fractal structure. They have been discovered and studied in many systems, the most prominent being the first order Lorenz and Rossler equations and the Henon map. Trajectories showing sensitive dependence on initial conditions in the infinite time limit are capable of displaying exponential separation on the average while remaining within the confines of the strange attractor. The starting point for this project was "What is...a Strange Attractor?" by David Ruelle, Notices of the American Mathematical Society. 53 (7): 764–765.

(Strange) Attractors

In the mathematical field of dynamical systems, an **attractor** is a set of numerical values toward which a system tends to evolve, for a wide variety of starting conditions of the system. A closed subset C of the phase space can be classified as an attractor if it possesses the following characteristics:

- C is *forward invariant.* i.e, if one point on a trajectory lies in C, then all subsequent iterations will remain in C.
- There exists a *basin of attraction* of which C is a subset such that all points lying in the basin will enter C in the limit of infinite time
- Finally, there exists no proper subset of C which fulfills the two preceding conditions.

A **Strange Attractor** is an attractor that has a fractal structure. A necessary condition for an attractor to classify as "strange" is that all trajectories that converge to it must exhibit sensitive dependence on initial conditions. This term was first coined by David Ruelle and Floris Takens in their 1971 paper describing fluid flow where they pointed out that turbulent flow results from a strange attractor regime in the Navier-Stokes equations. It is a relatively abstract mathematical concept which is recently witnessing applications in engineering and related systems.

Lorenz Attractor

For nearly 60 years, one of the classic icons of modern non-linear dynamics has been the Lorenz attractor. With its intriguing double-lobed shape and chaotic dynamics, the Lorenz attractor has symbolized order within chaos.

Edward Lorenz in the 1960s developed a system of three nonlinear differential equations inspired by a simplified model of convection rolls in the earth's atmosphere. These equations also describe the state of leaky water wheels and they also appear in descriptions of lasers and dynamos.

$$\dot{x} = a(y - x)$$
$$\dot{y} = x(b - z) - y$$
$$\dot{z} = xy - cz$$

a is known as the Prandtl number and b the Rayleigh number. The series does not form limit cycles nor does it ever reach a steady state. Instead it is an example of deterministic chaos. As with other chaotic systems the Lorenz system exhibits sensitive dependence on initial conditions, where two initial states will eventually display divergent trajectories, no matter how close they may be.

One normally assumes that the parameters a, b, c are positive. We explore the different possibilities of these values to find the strange attractor. The Jacobian for the system is:

$$\begin{pmatrix} -a & a & 0 \\ b-z & -1 & -x \\ y & x & -c \end{pmatrix}$$

The origin is always a fixed point. Its eigenvalues are

$$-c, \frac{-a-1-\sqrt{(a-1)^2+4ab}}{2}, \frac{-a-1-\sqrt{(a-1)^2+4ab}}{2}$$

We can see that the origin is stable in two directions for all positive values of a,b,c. When $b < 1$, all 3 directions are stable and all flows converge to the origin (global attractor). Hence a strange attractor cannot exist here since this violates sensitive dependence on initial conditions. A pitchfork bifurcation occurs at $b = 1$, and for $b > 1$ two additional critical points appear at:

$$(\sqrt{c(b-1)}\sqrt{c(b-1)}, b-1) \quad \text{and} \quad (-\sqrt{c(b-1)}, -\sqrt{c(b-1)}, b-1)$$

Continuing in this range, we wish to look at values where the points are repelling since otherwise, states wouldn't reach a chaotic attractor.. We see that a critical value of b exists beyond which these points become unstable. The calculations yield this value to be

$$b < a(\frac{a+c+3}{a-c-1})$$

which can hold only for positive b if $a > c+1$. The stability occurs through a subcritical Hopf bifurcation.

We take the standard values of $a = 10, c = 8/3, b = 28$ to satisfy the above conditions. By noting the properties that all volumes tend to zero and that the known fixed points are all repelling, we come to the conclusion that there must exist another attractor in this range of values. This is the strange attractor. In Figure 2, we have plotted the values of x against time for different initial values and after a transient, we see aperiodic behavior. Running simulations of the system on the R^3, we plotted the 3-dimensional view and the projections on the xz and yz planes (Figure 1)

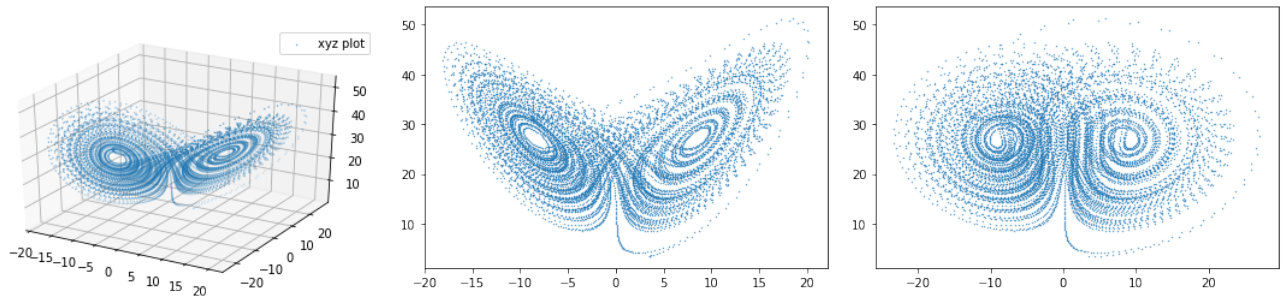

Figure 1: Lorenz attractor scatter plot, (3.5,3.5,3.5) [$a = 10, c = 8/3, b = 28$] (a) 3D view (b)XZ projection (C) YZ projection

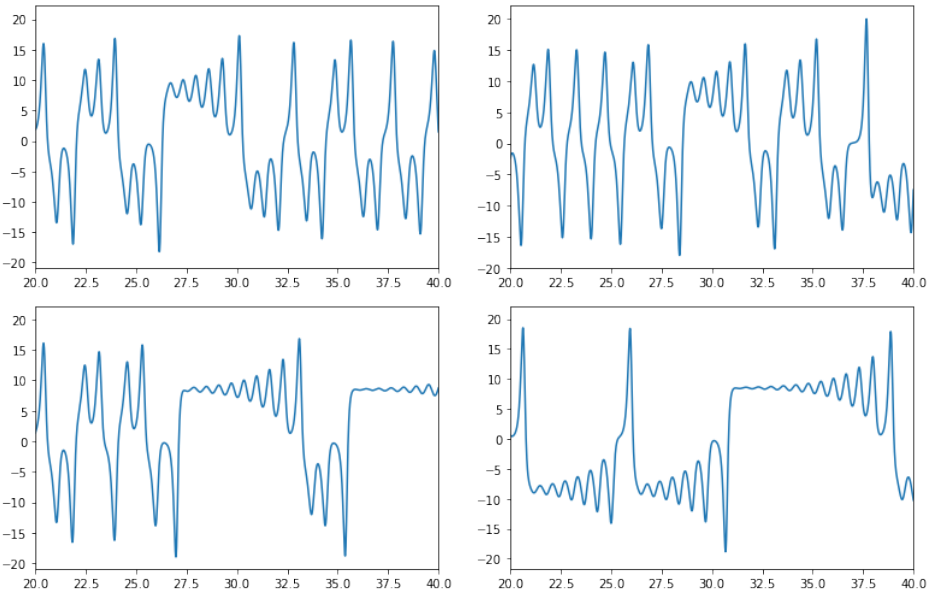

Figure 2: x vs t plots for Initial values $x_0 = y_0 = z_0 =$ (a) 3.3(top left) (b) 3.5(top right) (c) 3.7(bottom left) (d) 3.9(bottom right)

Rossler Attractor

The Rossler attractor is a chaotic attractor solution to the system

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

proposed by Otto Rossler (1976), often called Rossler system. Here, $(x, y, z) \in R^3$ are dynamical variables defining the phase space and $(a, b, c) \in R^3$ are parameters. Rossler had been motivated by the search for chemical chaos, that is, chaotic behavior in far-from-equilibrium chemical kinetics. To quote his inspiration:

> *"I wondered if a rope around the nose, circling it in several loops before falling off at the tip and then curving back to the starting point or its neighborhood, would not produce a similar tangle in 3D-space."*

The Rossler system is unique as it is minimal in chaos, for three reasons:

- It has a minimum phase space dimension of 3.
- It has a single non-linear term xz in the $\dot{z}$ equation.
- It generates a chaotic attractor with a single lobe as compared to the Lorenz attractor which has two lobes.

Trajectories in the Rossler attractor exhibit planar behaviour in the x-y plane as an outward spiral centered at a fixed point, and once they have reached a maximum radius from the center, the trajectory *jumps* in the z-direction and folds back onto the x-y plane to continue the subsequent motion. This behaviour is the simplest explanation of making evident the requirement that chaotic flow requires more than two dimensions.

The fixed points for the Rossler system are recovered by setting each of the equations to zero, and solving for the three unknowns x, y and z.

We get the two fixed points as:

$$x = \frac{c - \sqrt{c^2 - 4ab}}{2}, y = \frac{-c + \sqrt{c^2 - 4ab}}{2a}, z = \frac{c - \sqrt{c^2 - 4ab}}{2a}$$

and

$$x = \frac{c + \sqrt{c^2 - 4ab}}{2}, y = \frac{-c - \sqrt{c^2 - 4ab}}{2a}, z = \frac{c + \sqrt{c^2 - 4ab}}{2a}$$

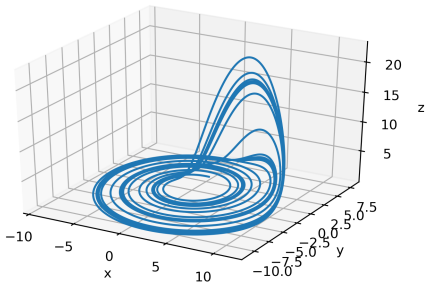

Figure 3: 3D figure of a Rossler attractor for a = 0.2, b = 0.2, c = 5.7

On solving for the parameter values a = 0.2, b = 0.2, c = 5.7 we obtain the fixed points as

$$(0.007, -0.035, 0.035) \quad \text{and} \quad (5.69, -28.5, 28.5)$$

It has been noted in prior studies of this attractor that the latter set of fixed points is insignificant, as the initial conditions in its vicinity tend to be pushed towards the fixed point near origin. Hence, we will focus our analysis on the former set of fixed points.

Solving for the eigenvalues of the Rossler attractor, we first obtain the Jacobian matrix as follows

$$\mathrm{J} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & a & 0 \\ z & 0 & x - c \end{pmatrix}$$

The cubic equation obtained from solving $|J - \lambda I|$ is

$$\lambda^3 + (a + x - c)\lambda^2 + (ax + 1 + z - ac)\lambda - x - az + c = 0$$

Taking the fixed point closest to origin $(0.007, -0.035, 0.035)$ and the parameters $(a = 0.2, b = 0.2, c = 0.2)$ the cubic equation can be solved to yield the following results for λ:

$$\lambda_{1,2} = 0.097012 \pm 0.99519i, \lambda_3 = -5.687024$$

From the obtained eigenvalues we can infer that the first two eigenvalues correspond to the outward circular motion on the x-y plane while the third eigenvalue being negative corresponds to the attracting influence or "reinjection" of the system along the z direction.

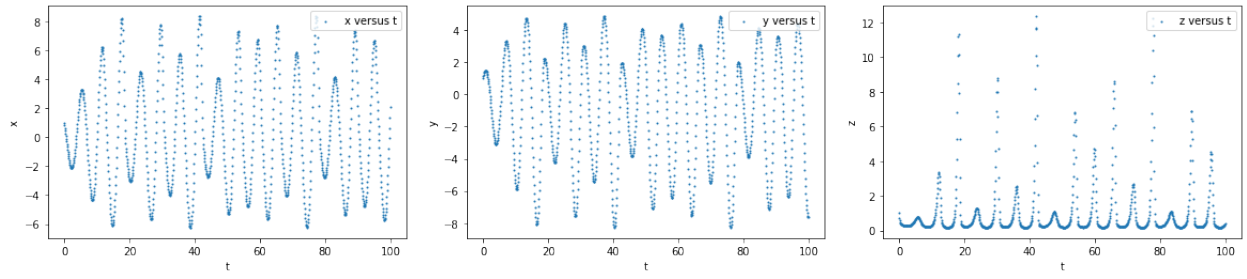

Figure 4: (a) x vs t, (b) y vs t, (c) z vs t plots for initial conditions $x_0 = y_0 = z_0 = 1$ for the Rossler system

Henon Map

The Henon map is a discrete-time system, which takes a point (x_n, y_n) in the plane and maps it to a new point

$$x_{n+1} = 1 + y_n - ax_n^2$$

$$y_{n+1} = bx_n$$

This map can be looked upon as a series of transformations on a plane of initial points within the basin of attraction in the following manner:

- Folding and stretching

$$x' = x, y' = 1 + y - ax^2$$

- Compression along x axis

$$x'' = bx', y'' = y'$$

- Reflection along the line y = x

$$x''' = y'', y''' = x''$$

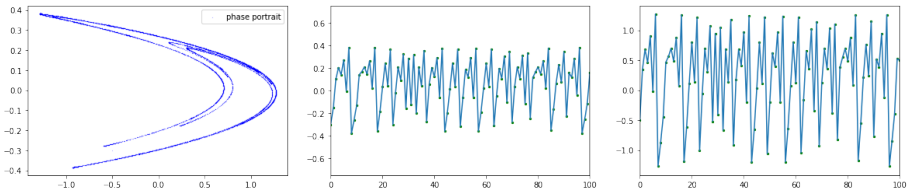

Figure 5: (a)Strange attractor in the Phase portrait of the Henon map (b) y vs t (c) x vs t [$a = 1.4, b = 0.3$, starting at $(1, 1)$]

We have plotted the phase potrait of the Henon map for the initial values $(1, 1)$ with the classical value of parameters, $a = 1.4, b = 0.3$. For the same parameter and initial value, we have also plotted the x vs t and y vs t graphs of the Henon map. It is evident from them that we have a chaotic strange attractor for these values.

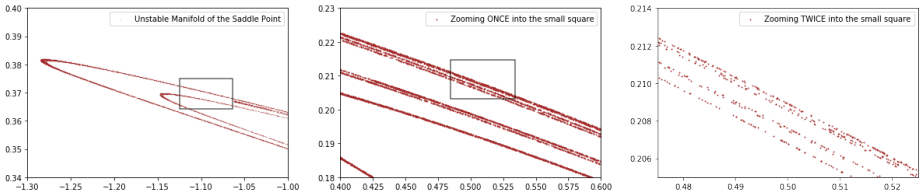

Figure 6: Changing view scales of the saddle/strange attractor of the Henon map

Consider the boxed part of the Henon map (Figure 6): It appears smooth in one direction and acts as a Cantor set (which is a fractal itself) in another direction. In the transverse direction, we see that the Henon map has 6 groups of parallel curves. Reducing the scale, we see that each group of curves contains finer groups of 6 lines and the self similarity becomes apparent. This continues ad infinitum. Hence the fractal structure of the strange attractor becomes apparent.

Fractal Dimension

There is no clear definition of fractals. However, two important characteristics of fractals are

- having some degree of self-similarity and
- that they have detailed structures at arbitrary scales (as opposed to becoming featureless and smooth at small scales).

A fractal dimension is an index for characterizing fractal patterns or sets by quantifying their complexity as a ratio of the change in detail to the change in scale. In general, a more complex fractal object has a higher fractal dimension. There are different measures of this, notably the box, the correlation, and the Hausdorff dimension.

Box Dimension

To calculate the box dimension for a fractal, we imagine the fractal lies on an evenly spaced grid (of squares, cubes or hypercubes, accordingly from the dimension of the topological space that the fractal lies in). The box-counting dimension is calculated by observing how the number of squares or cubes covering the fractal structure changes as we make the grid finer, by applying a box-counting algorithm. Suppose $N(\epsilon)$ is the number of boxes of side length ϵ required to cover the fractal. Then the box-counting dimension is defined as:

$$dim_{box} := \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log (1/\epsilon)}$$

Roughly, we are trying to correspond the surface area, volume or hyper-volume of the fractal as a power law, which is what one expects in the case of a smooth manifold of integer dimension d.

$$N(\frac{1}{n}) \approx Cn^d$$

e.g. area of circle $= \pi r^2$, this indeed matches up with our expectation that a circle is a 2-dimensional object.

Hausdorff Dimension

The Hausdorff dimension is a commonly used definition for measuring the fractal dimension of a strange attractor. The formal definition of the Hausdorff dimension is as follows:

Consider a fractal lying in a space with topological dimension n. For every $d > 0$ there exists a (atmost) countable closed cover of the fractal, with sets having diameter less than d. We define the Hausdorff measure of a fractal (of α dimensions) to be the infimum of a set of positive numbers, each being greater than the sum of these diameters raised to the power α. The Hausdorff dimension is the infimum of the set of α having a corresponding Hausdorff measure 0. Informally, instead of using hypercubes of fixed sizes as in the box dimension, we have used closed sets of varying sizes.

The calculation of the box and Hausdorff dimensions is quite involved, and hence is out of the scope of this report.

Conclusion

Even seemingly simple dynamical systems can exhibit a wide range of exciting properties and complex behavior by varying the value of its parameters. This is most profoundly exhibited by the existence of a strange attractor in the single non-linearity wielding Rossler system. These dynamical systems still have vast areas unexplored along parameter space. We found it fascinating that flows which cannot intersect each other in autonomous systems and are smooth everywhere can give rise to bounded attractors that occupy zero volume without periodicity and are not fixed points or limit cycles. We recognize that this is possible only due to the fractal nature of these objects as they can show fine structure at any scales.

Bibliography

The following resources have been indispensable to our research and in completing this report.

1. Ruelle D., *Strange Attractors*
2. Strogatz S. (1994), *Nonlinear Dynamics and Chaos (Perseus Books)*
3. Ott, E. "Appendix: Hausdorff Dimension." *Chaos in Dynamical Systems. New York: Cambridge University Press, pp. 100-103, 1993*
4. D. Auerbach, P. Cvitanovic, J.-P. Eckmann, G.H. Gunaratne, I. Procaccia, *Exploring chaotic motion through periodic orbits, Phys. Rev. Lett. 58 (1987) 2387.*
5. Kenneth Falconer (1990), *Fractal Geometry: Mathematical Foundations and Applications (New York: Wiley)*

Appendix- Codes

Lorenz Attractor

```
#!/usr/bin/env python
# coding: utf-8

# In[39]:

import numpy as np
from matplotlib import pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
from mpl_toolkits.mplot3d import Axes3D

# Parameters

# In[40]:

a = 10
b = 28
c = 8/3

# Initial Conditions

# In[41]:

x0 = 3.5
y0 = 3.5
z0 = 3.5

dt = 0.01
t0 = 0

# Iterative Algorithm

# In[42]:

x = x0
y = y0
z = z0
t = t0

x_list = []
y_list = []
z_list = []
t_list = []

for i in range(10000):
    x_list.append(x)
    y_list.append(y)
    z_list.append(z)
    t_list.append(t)

    x1 = x + a*(y-x)*dt
    y1 = y + (x*(b-z)-y)*dt
    z1 = z + (x*y-c*z)*dt
    t += dt

    x = x1
    y = y1
    z = z1

# In[51]:

plt.plot(t_list, x_list)
plt.xlim(20,40)

```

```
# In[50]:

plt.plot(t_list, y_list)
plt.xlim(20,40)

# In[49]:

plt.plot(t_list, z_list)
plt.xlim(20,40)

# In[52]:

plt.scatter(x_list, y_list, s=0.1)

# In[53]:

plt.scatter(x_list, z_list, s=0.1)

# In[57]:

plt.scatter(y_list, z_list, s=0.1)

# In[38]:

plt.figure()

ax = plt.axes(projection='3d')
ax.scatter(np.array(x_list), np.array(y_list), np.array(z_list), label="xyz plot", s=0.1)
plt.legend(loc='best')

# In[ ]:
```

Rossler Attractor

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
from mpl_toolkits.mplot3d import Axes3D

# Initial conditions
# In[2]:
x0 = 1
y0 = 1
z0 = 1

# Parameters
# In[3]:
a = 0.2
b = 0.2
c = 5.7

t0 = 0
dt = 0.1

# Plotting
# In[4]:
x_list = []
y_list = []
z_list = []
t_list = []
```

```
35
36 x1 = x0
37 y1 = y0
38 z1 = z0
39 t1 = t0
40
41 for extent in range(100000):
42     x_list.append(x1)
43     y_list.append(y1)
44     z_list.append(z1)
45     t_list.append(t1)
46
47     x2 = x1 - (y1 + z1)*dt
48     y2 = y1 + (x1 + a*y1)*dt
49     z2 = z1 + b + z1*(x1 - c)*dt
50     t2 = t1 + dt
51
52     x1 = x2
53     y1 = y2
54     z1 = z2
55     t1 = t2
56
57 plt.figure()
58 plt.scatter(x_list, y_list, s=0.01, label='xy plot')
59 plt.xlabel('x')
60 plt.ylabel('y')
61 plt.legend(loc='best')
62
63 plt.figure()
64 plt.scatter(x_list, z_list, s=0.01, label='xz plot')
65 plt.xlabel('x')
66 plt.ylabel('z')
67 plt.legend(loc='best')
68
69 plt.figure()
70 plt.scatter(y_list, z_list, s=0.01, label='yz plot')
71 plt.xlabel('y')
72 plt.ylabel('z')
73 plt.legend(loc='best')
74
75 plt.figure()
76 ax = plt.axes(projection='3d')
77 ax.scatter3D(np.array(x_list), np.array(y_list), np.array(z_list), label="xyz plot", s=0.01)
78 ax.set_xlabel('x')
79 ax.set_ylabel('y')
80 ax.set_zlabel('z')
81 plt.legend(loc='best')
82 # ## Time-series plots
83 # In[5]:
84 plt.figure()
85 plt.scatter(t_list[:1000], x_list[:1000], s=1, label="x versus t")
86 plt.xlabel("t")
87 plt.ylabel("x")
88 plt.legend(loc='upper right')
89
90 plt.figure()
91 plt.scatter(t_list[:1000], y_list[:1000], s=1, label="y versus t")
92 plt.xlabel("t")
93 plt.ylabel("y")
94 plt.legend(loc='upper right')
95
96 plt.figure()
97 plt.scatter(t_list[:1000], z_list[:1000], s=1, label="z versus t")
98 plt.xlabel("t")
99 plt.ylabel("z")
100 plt.legend(loc='upper right')
101 # ## Rossler Map for $x$
102 # In[6]:
```

```
xmax_list = []
for i in range(1,len(x_list)-1):
    x = x_list[i]
    if (x > x_list[i-1] and x > x_list[i+1]):
        xmax_list.append(x)
plt.scatter(xmax_list[:-1], xmax_list[1:], s=0.1)
plt.xlim(0,14)
plt.ylim(0,14)

# ## Visualizing Chaos in the XY plot
# In[10]:
def chaos_plot(c):
    x_list = []
    y_list = []
    z_list = []
    t_list = []

    x1 = x0
    y1 = y0
    z1 = z0
    t1 = t0

    for extent in range(10000):
        x_list.append(x1)
        y_list.append(y1)
        z_list.append(z1)
        t_list.append(t1)

        x2 = x1 - (y1 + z1)*dt
        y2 = y1 + (x1 + a*y1)*dt
        z2 = z1 + b + z1*(x1 - c)*dt
        t2 = t1 + dt

        x1 = x2
        y1 = y2
        z1 = z2
        t1 = t2

    plt.figure()
    plt.scatter(x_list, y_list, s=0.1, label='xy plot')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend(loc='best')
    plt.xlim(-10,10)
    plt.ylim(-10,10)

interact(chaos_plot, c=widgets.FloatSlider(min=1.2,max=10,step=0.1,value=5.7))
```

Henon Map

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
# Initial Conditions
# In[2]:
x0 = 1
y0 = 1

# Parameters
# In[3]:
a = 1.4
```

```
b = 0.3

t0 = 0
dt = 1

# Plotting the Phase Portrait
# In[4]:
x_list = []
y_list = []
t_list = []

x0 = -0.5
y0 = -0.3
t = t0

for t_range in range(10000):
    x_list.append(x0)
    y_list.append(y0)
    t_list.append(t)

    x = y0 + 1 - a*(x0**2)
    y = b*x0
    x0 = x
    y0 = y
    t += dt

plt.scatter(x_list, y_list, c='blue', s=0.01, label='phase portrait')
plt.legend()
# Interactive Phase Portrait
# In[19]:
def phase_portrait(a, b, t0=0, dt=1):

    x_list = []
    y_list = []
    t_list = []

    x0 = -0.5
    y0 = -0.3
    t = t0

    for t_range in range(10000):
        x_list.append(x0)
        y_list.append(y0)
        t_list.append(t)

        x = y0 + 1 - a*(x0**2)
        y = b*x0
        x0 = x
        y0 = y
        t += dt

    plt.scatter(x_list, y_list, c='blue', s=0.01, label='phase portrait')
    plt.legend()

interact(phase_portrait, a=widgets.FloatSlider(min=1,max=1.45,step=0.01,value=1.4), b=widgets.
    FloatSlider(min=0,max=0.4,step=0.01,value=0.3))
# ### Unstable Manifold of the Saddle Point
# In[6]:
x_list = []
y_list = []
t_list = []

x0 = -0.5
y0 = -0.3
t = t0

for t_range in range(100000):
    x_list.append(x0)
```

```
    y_list.append(y0)
    t_list.append(t)

    x = y0 + 1 - a*(x0**2)
    y = b*x0
    x0 = x
    y0 = y
    t += dt

plt.scatter(x_list, y_list, c='brown', s=0.01, label='Unstable Manifold of the Saddle Point')
plt.legend()
plt.xlim(-1.3,-1)
plt.ylim(0.34,0.4)

# In[7]:
plt.scatter(x_list, y_list, c='brown', s=0.7, label='Zooming ONCE into the small square')
plt.legend()
plt.xlim(0.4, 0.6)
plt.ylim(0.18, 0.23)
plt.figure()
plt.scatter(x_list, y_list, c='brown', s=0.7, label='Zooming TWICE into the small square')
plt.legend()
plt.xlim(0.475, 0.525)
plt.ylim(0.205, 0.214)
# ### Plot of x(t) versus t
# In[17]:
plt.scatter(t_list, x_list, c='green', s=5)
plt.plot(t_list, x_list)
plt.xlim(0,100)

# ### Plot of y(t) versus t
# In[16]:
plt.scatter(t_list, y_list, c='green', s=5)
plt.plot(t_list, y_list)
plt.xlim(0,100)
plt.ylim(-0.75, 0.75)
```

Japanese/Ueda Map

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
from mpl_toolkits.mplot3d import Axes3D


# Parameters

# In[2]:


k = 0.1
B = 7.5
delta_t = 1e-3


# Inital Conditions

# In[3]:

```

```
30
31 x_init = 3
32 y_init = 2
33
34
35 # Iterative Algorithm
36
37 # In[6]:
38
39
40 x_init = 2
41 y_init = 2
42
43 x_n = x_init
44 x_n_plus_1 = (y_init)*(delta_t) + (x_init)
45 t_n = 0
46
47 t_counter = []
48 x_counter = []
49 y_counter = []
50
51 for iter_count in range(100000):
52     t_n = (iter_count)*(delta_t) ; t_counter.append(t_n)
53     y_n = (x_n_plus_1 - x_n)/delta_t ; y_counter.append(y_n)
54     x_counter.append(x_n)
55     x_n_plus_2 = 2*(x_n_plus_1) - (x_n) + ((delta_t)**2)*(B*(np.cos(t_n)) - (x_n)**3) - ((k)*(
       delta_t))*((x_n_plus_1) - (x_n))
56     x_n = x_n_plus_1
57     x_n_plus_1 = x_n_plus_2
58
59 plt.scatter(x_counter, y_counter,c='green',label="Phase Portrait", s=0.01)
60 plt.legend(loc='best')
61
62 plt.figure()
63
64 y_2npi = []
65 x_2npi = []
66 t_2npi = []
67 for i in range(len(t_counter) - 1):
68     if abs((y_counter[i+1]-y_counter[i])/delta_t + k*(x_counter[i+1]-x_counter[i])/delta_t + (
       x_counter[i])**3 - B) < 0.1:
69         y_2npi.append(y_counter[i])
70         x_2npi.append(x_counter[i])
71         t_2npi.append(t_counter[i])
72 plt.scatter(x_2npi, y_2npi,c='blue', label="Stroboscopic Map", s=0.01)
73 plt.legend(loc='best')
74
75 plt.figure()
76
77 ax = plt.axes(projection='3d')
78
79 ax.plot3D(np.array(x_counter), np.array(y_counter), np.array(t_counter), label="xyt plot")
80 plt.legend(loc='best')
81
82
83 # Interactive plot
84
85 # In[8]:
86
87
88 def plots(B):
89
90     x_init = 2
91     y_init = 2
92
93     x_n = x_init
94     x_n_plus_1 = (y_init)*(delta_t) + (x_init)
95     t_n = 0
```

```
96
97      t_counter = []
98      x_counter = []
99      y_counter = []
100
101     for iter_count in range(30000):
102         t_n = (iter_count)*(delta_t) ; t_counter.append(t_n)
103         y_n = (x_n_plus_1 - x_n)/delta_t ; y_counter.append(y_n)
104         x_counter.append(x_n)
105         x_n_plus_2 = 2*(x_n_plus_1) - (x_n) + ((delta_t)**2)*(B*(np.cos(t_n)) - (x_n)**3) - ((k)*(
    delta_t))*((x_n_plus_1) - (x_n))
106         x_n = x_n_plus_1
107         x_n_plus_1 = x_n_plus_2
108
109     plt.scatter(x_counter, y_counter,c='green',label="Phase Portrait", s=0.01)
110     plt.legend(loc='best')
111
112 interact(plots, B=widgets.FloatSlider(min=0,max=20,step=0.5,value=10))
113
114
115 # In[ ]:
```