Exploring Chaos

Henry Pacheco Cachon*

Colby College

Waterville, Me

(Dated: 2 December 2020)

* Author's email: hpache22@colby.edu

I. INTRODUCTION: THE HISTORY OF CHAOS

Chaos Theory is a fairly new field in physics and mathematics, which aims at studying the randomness found within deterministic laws with different initial conditions. In other words, the people within this field study laws and equations that are extremely sensitive to initial conditions, which over a period of time, the outputs of these equations are nearly impossible to predict just from the initial conditions.

Chaos Theory can be traced back as early as the 17th century. The first time people had an interest in Chaos, happened with Isaac Newton and his equations for the motions of the planets, specifically his equations which dealt with the orbits of 3 gravitational bodies in space.[1] What Newton found, was that his equations were too simplified, which led to wrong predictions. A couple years later, Newton realized that the simplification that he made to his models left out the gravitational interactions that each body had, which is why his models did not predict the motion of three bodies very well.

At this point in history, people had a great grasp in predicting the orbits of two gravitational bodies, but there was a general consensus within the community, that solving the three-body problem and generalizing that solution to n-bodies is one of the most important problems of their time. This led to King Oscar II to propose a prize to anyone who can find a model for the n-body problem (the actual task was to predict the orbit of our solar system 10 million years in the future). As we know now, no such model has been discovered, but someone still got a prize, and that was Henri Poincaré, which the paper will discuss later. Unknown to Poincaré and others, this was the birth of Chaos Theory, since his solution involved studying the phase space of a chaotic system.[1]

Chaos Theory did not blow up after Poincaré though, there were many mathematicians and physicists that built on Poincaré's work, such as Kolmogorov's work which actually resulted in a theorem that describes the movement of the n-body problem towards chaos, but that also never caught on. Chaos Theory would only catch on after Lorenz rediscovered Chaos within his simplified model of the weather, which gave birth to a wonderful field of research that has its applications in biology, epidemiology, orbital mechanics, and so on.

II. POINCARÉ AND THE N-BODY PROBLEM

As stated in the introduction, Poincaré's work with the n-body problem is seen as the birth of Chaos Theory. In Poincaré's paper "On the three body problem and the equations of the dynamics," he did not find an analytical solution to the n-body problem, but he did use a generalized form of the solution to the 3 body problem in order to explore the n-body problem. What Poincaré found was three types of solutions, the first was what he called a periodic solution, the second was a asymptotic solution, and the third was a doubly asymptotic solution.[2] A full disclaimer before moving on, these solutions are not analytical solutions, they are numerical solutions.

Although exploring the n-body problem and Poincaré's solution to that problem is very interesting, that is not the main focus of this paper, so let's skip to the part of Poincaré's solution which is not only the birth of Chaos Theory, but also the birth of a very useful tool when it comes to studying dynamical systems, Poincaré sections. Essentially, in his attempt to predict the trajectory of a gravitational body over a long period of time, Poincaré proposed the use of a plane within phase space in order to see where the intersections of the phase trajectory trended towards. The way this method worked for the n-body problem was that you would first numerically solve the n-body equations over a long period of time, you would then plot the phase trajectory, and cut a part of that trajectory with a plane. After doing that, you would then plot the points which intersect with that plane. This resulted in the following figure:

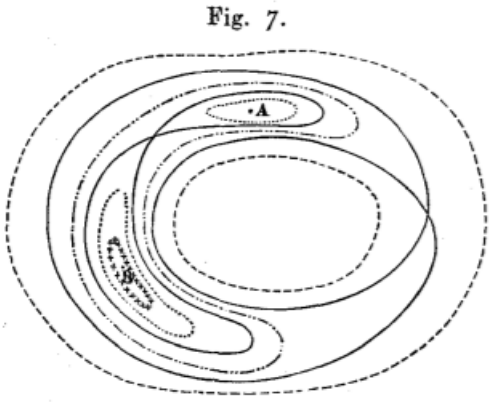

FIG. 1. Poincaré map found in "On the three body problem and the equations of the dynamics."

The Poincaré Section, is a very useful tool when studying the phase space of a chaotic system. The reason behind this is because this allows people to study the stability of a phase trajectory which gives people insight into the overall stability of a dynamical system. In the case of the n-body problem, Poincaré's use of this method actually showed that the motion of the planets were very sensitive to initial conditions and therefore exhibit chaotic behavior.[1]

III. POINCARÉ SECTIONS AND THE DUFFING OSCILLATOR

The idea that the n-body problem is really a problem about a chaotic dynamical system went unnoticed for a long time, but the use of Poincaré sections became widespread once people started studying chaotic dynamical systems. There are a lot of well known chaotic systems such as the double pendulum, but one of the systems which results in an interesting Poincaré section is known as the Duffing Oscillator.

The Duffing Oscillator, is just a type of differential equation that describes the motion of a family of driven or damped oscillators. This differential equation was made by Georg Duffing, and his goal was to make a model for oscillators that have a potential that does not follow Hooke's law.[3] The equation is the following:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma cos(\omega t) \tag{1}$$

Depending on your choices for δ, α, β, γ, and ω, the system can either be in a chaotic state or in a stable state. In order to solve the Duffing equation with Matlab, you would need to change the equation into a system of first order ODE's. This can be done easily by first solving for $\ddot{x}$ in equation 1:

$$\begin{aligned} \ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 &= \gamma cos(\omega t) \\ \ddot{x} &= -\delta\dot{x} - \alpha x - \beta x^3 + \gamma cos(\omega t) \end{aligned}$$

We now let $\dot{x} = y$ to get the following coupled first ordered differential equations:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= -\delta y - \alpha x - \beta x^3 + \gamma cos(\omega t) \end{aligned} \tag{2}$$

Now that we have a system of equations to model, we can use matlab (or any other program) to numerically solve these equations, but you may notice that we only have two equations and not three, which is an issue if we want to use a Poincaré section. We can add a third equation, $z = \omega$, but we will use a Poincaré map instead. The general idea for a Poincaré map is the same, but instead of using a plane to pick out points, we will pick out every nth point in our numerical solutions. The rule to pick out every nth point will be to pick a point every T seconds, in which T is the period of the driving force in the Duffing Oscillator. This rule may seem extremely random, but I am planning on changing the amplitude of the driving force, so it makes sense to pick the period of the driving force as a rule to construct our map.

Now that we have everything all set up, we can set our parameters to be $\delta = 0.3$, $\alpha = 1$, $\beta = 1$, $\omega = 1.2$. We are only going to look at two values for *gamma* since both these values exhibit chaotic and periodic behavior. The first is $\gamma = 0.37$ which exhibits periodic behavior and $\gamma = 0.50$ which exhibits chaotic behavior.[3]. The reason I chose these two values is to demonstrate how powerful Poincaré sections are when it comes to studying dynamical systems.

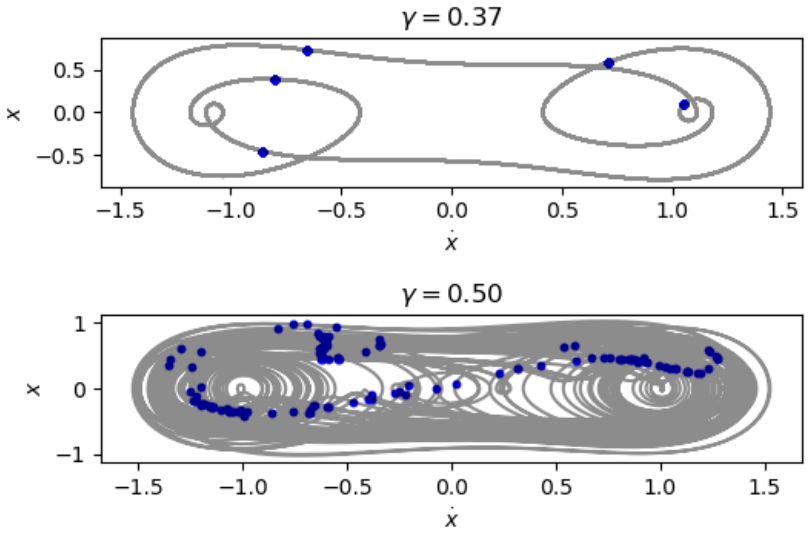

FIG. 2. Phase trajectories and Poincaré maps of Duffing Oscillator with different γ values. This was made with python

From figure 2, we can see how the phase trajectories for both values of gamma differ. For starters, the phase trajectory for $\gamma = 0.37$ is stable and it only has one trajectory whereas

the one for $\gamma = 0.50$ is all jumbled up and never repeats itself. The Poincaré maps also have a striking difference; for $\gamma = 0.37$, there are only 5 points, and for $\gamma = 0.50$, there are multiple points making a weird shape. These plots were made with a limited time span, but if we expand that time span to a large number like $t = 18000$, and only focus on the Poincaré maps, we get the following figures.

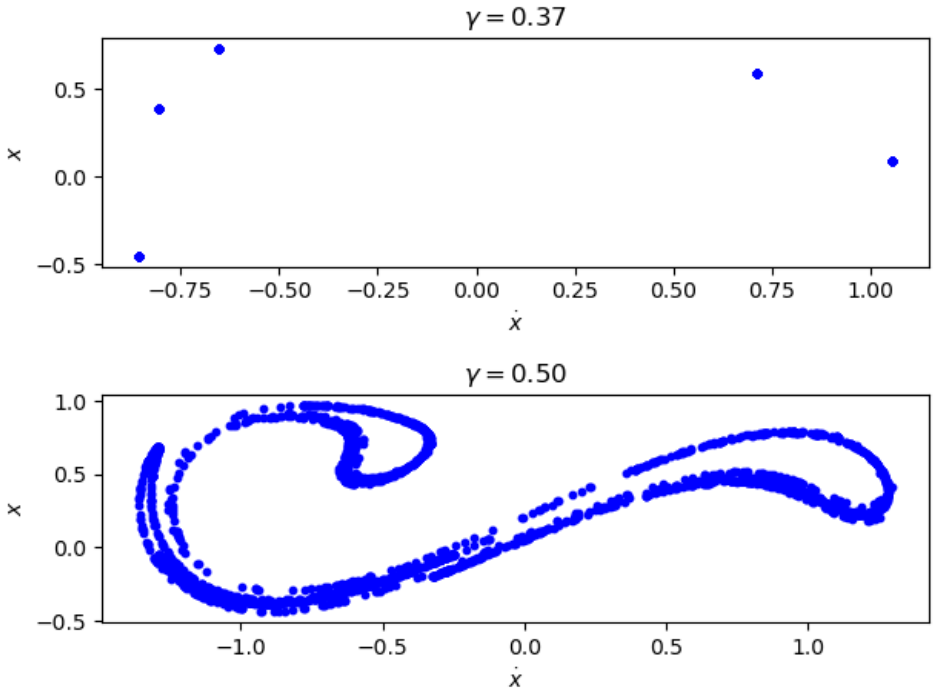

FIG. 3. Plot of the Poincaré maps made in python.

Here we can now clearly see the differences in the Poincaré maps for both values of γ. The first thing to note is that if we have a normal periodic oscillator, the Poincaré map will only have points. In this case, we have five points because for $\gamma = 0.37$, the Duffing Oscillator is inside of a five-period bifurcation. [3] For non-periodic oscillators, the Poincaré map wont be a limited number of points, but instead it will be some sort of weird static shape. To further drive this dichotomy, if we were to repeat this for a simple pendulum, we would only get one point on the Poincaré map. The other important thing to note before moving on is that both figures 1 and 3 are extremely cleaned up. These plots for $\gamma = 0.37$ are made using points after it reaches a limit cycle, if we were to look at the system from $t = 0$, we would notice that there would be more than five points and that these points are also converging to the limit cycle.

IV. LORENZ EQUATIONS

Now, the main focus will be on the Lorenz Equations, from Edward Lorenz's paper, "Deterministic Nonperiodic Flow." These equations were derived from a series of fluid dynamics equations which Lorenz used to model the weather on a computer. The story goes that Lorenz was modeling the weather in his Royal McBee computer, and he wanted to remake a weather pattern that he saw earlier. The way he did this was by starting the new simulation midway through the original simulation, but what he noticed was that even if he used the same conditions, the two patterns diverged in an unpredictable manner.[1] He initially thought that something in his computer broke, but after checking every single part, he realized that the issue was within the initial conditions and not his computer. This caused Lorenz to simplify the fluid dynamic equations into a system of three differential equations of the form [4]

$$\begin{aligned} \dot{x} &= -\sigma(x-y) \\ \dot{y} &= -xz + rx - y \\ \dot{z} &= xy - bz \end{aligned} \tag{3}$$

The outputs for x, y, z are pretty interesting to plot, but the main focus of the study of Lorenz's equations, and also the main focus of Lorenz's paper is the phase space plots that you get from these equations. Traditionally, the model parameters are set to be $\sigma = 10$ and $b = 8/3$, with initial conditions $x(0) = 0, y(0) = 0.001, z(0) = 0$. The choice to make r a varying parameter is not a random choice, because under Lorenz's simplifications, r is just a value proportional to the Rayleigh number.

Since we have a system of three coupled ODEs, we can use Poincaré sections to study these equations. The most common Poincaré section that is examined in terms of the Lorenz equations is a plane parallel to the xy plane. The location of this plane is arbitrary, but the most common point is at $z = r$.

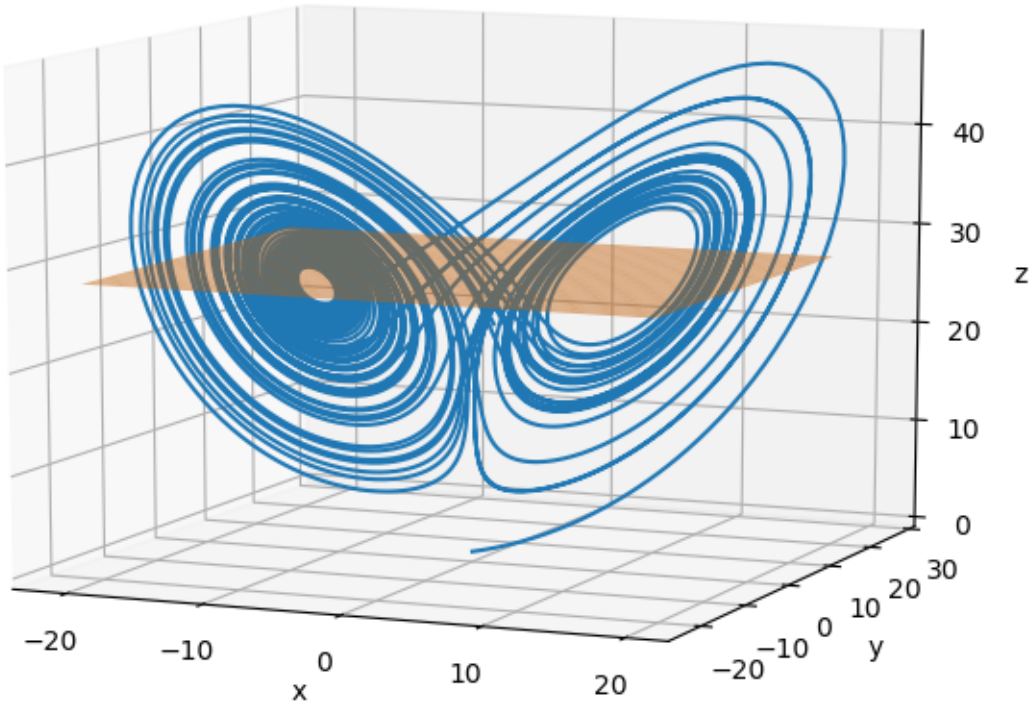

FIG. 4. Phase trajectory at r = 28 with our Poincaré section represented by the orange plane. This was made in python.

Here, the plane is transparent, but we can see that our phase trajectory clearly passes through this plane at multiple points. We can use this section to make a Poincaré map, which will specifically be on the xy projection of our phase trajectory. As stated in the previous sections, chaotic dynamical systems have a Poincaré map that make a weird shape instead of only having a limited number of points, and this is something that we are going to observe since the Lorenz equations begin to exhibit chaotic behavior at $r > 14$.

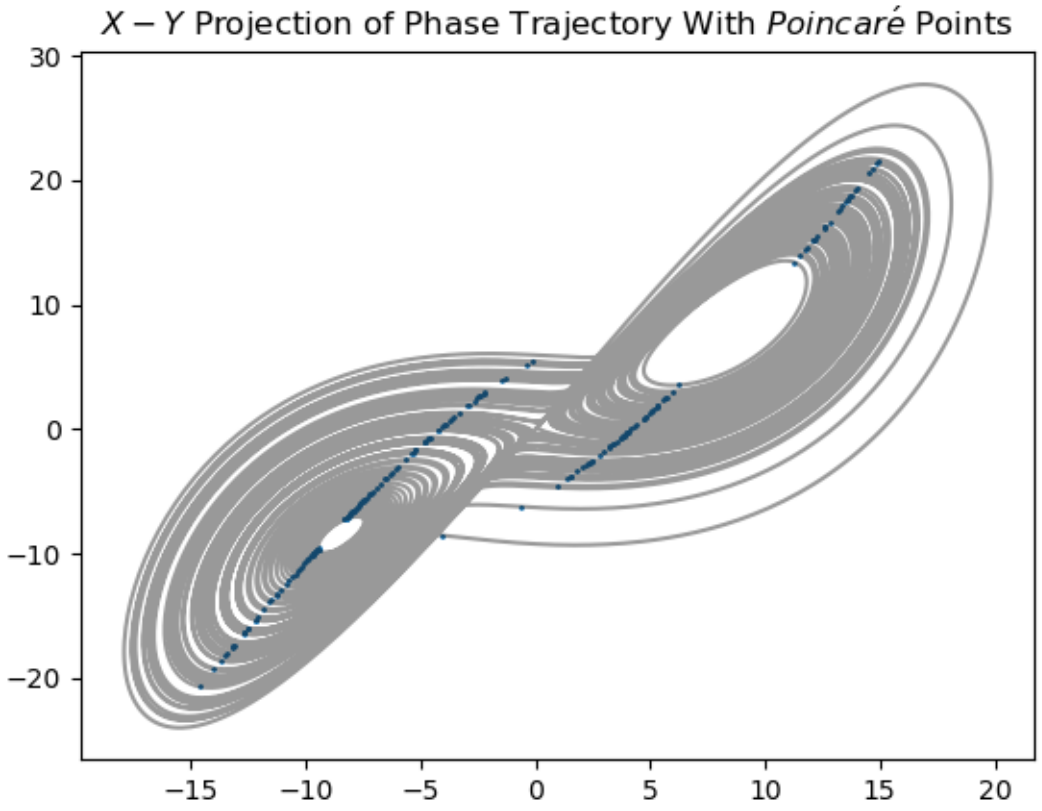

FIG. 5. Superimposed plot for a Poincaré section at $r = 28$. The blue scatter plot is the Poincaré map and the gray plot is the projection of the phase trajectory onto the xy plane. Made with python

Here, we have figure 5, which shows us the phase trajectory projected onto the xy plane and the Poincaré map that we got from our Poincaré section at $z = 28$. Notice that the blue dots are making a shape and it seems like they are converging to two points outside of the trajectory, but that is not the case. If we run our numerical solver for a longer time, we will see that the points on the Poincaré map never converges to anything as seen in figure 6.

We can continue making Poincaré maps for different values of r, but that is not an efficient way to explore the Lorenz equations. If we wanted to determine if the Lorenz equations ever exhibit periodicity for values of $r > 14$, we may need to sift through a lot of values of r before we come across any periodicity, assuming there is periodicity to begin with. We can get around this problem by making a bifurcation diagram with our Poincaré sections.

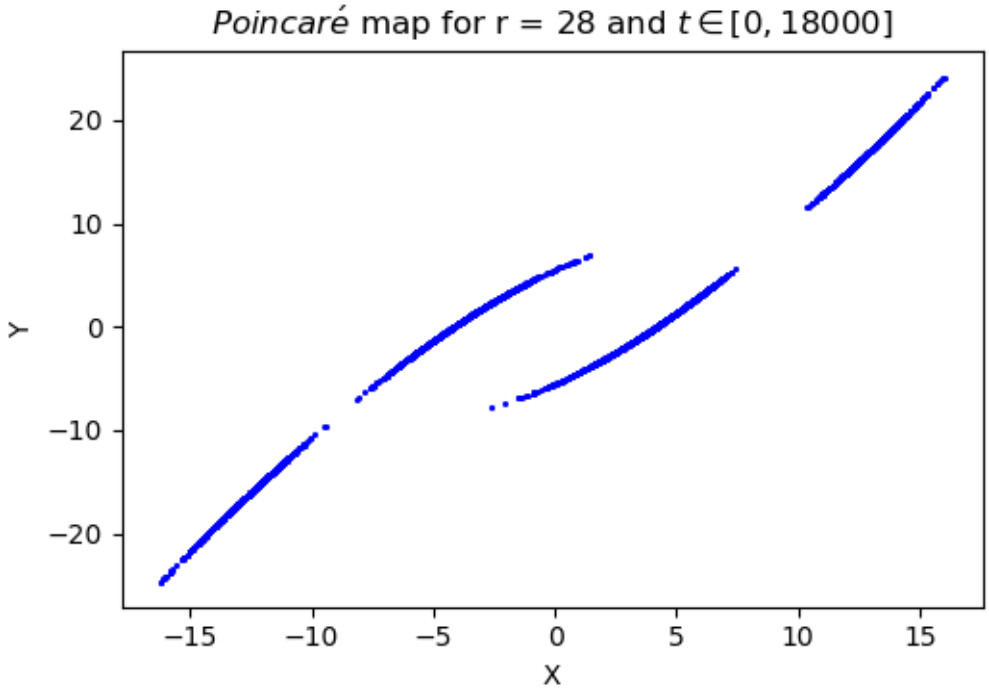

FIG. 6. Poincaré map for r=28 ran for 18000 time units. Made with python

The way we'll make this bifurcation diagram is to first solve the Lorenz equations after 50 time units, and then go through our solutions until we have a z value that is within $z \in [r-1, r]$. We then extract all the x and y points that corresponds to that z value. From there we can choose to extract one x or y point and then plot it against its respective r value.

A couple of things to note is that picking to focus only on x or y does result in a different looking bifurcation diagram. They will have the same characteristics, but x will need more r points in order to make those characteristics clear. The other thing to note is that the way in which we choose our one point will also affect the bifurcation diagram. Choosing the largest or smallest value will result in a one sided bifurcation diagram, and chosing the average or median of all the points will result in a spread out diagram. I decided to plot the largest, smallest, average, and last points in the y direction and then made one large bifurcation diagram in order to get the most detail out of it. This does take a ton of computing time, but the payoff was worth it.

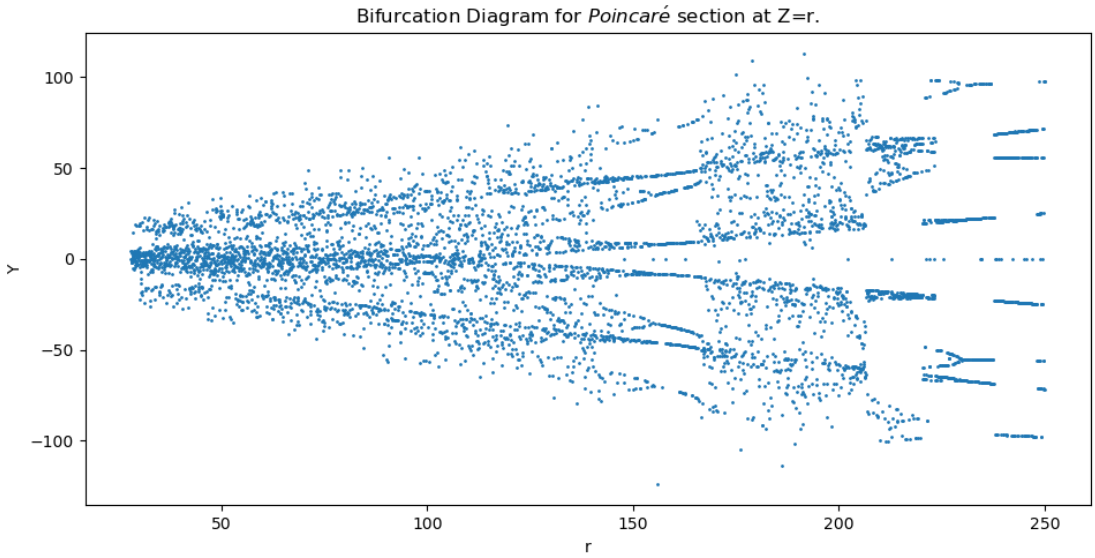

FIG. 7. Bifurcation diagram for Poincaré section at $Z = r$. Made with python

If you look closely at figure 7, we see two different regions which behave differently. One region has points scattered all over the place with no sense of order, this region is the chaotic region for the Lorenz equations. In other words, we can expect the r values in these regions to exhibit non-periodicity within their solutions. The second region has points making some branching structure, this region is known as a period doubling bifurcation, in which we can expect our solutions to oscillate about a limited number of points. This means that for values of r in these regions, we can expect our solutions to exhibit periodicity. We can confirm this by picking values of r in these regions, and plotting $Y(t)$ to see if there really is periodicity. The values that I picked are $r = 100$, $r = 155$, and $r = 245$. According to the bifurcation diagram, $r = 100$ will be chaotic and $r = 155$ and $r = 245$ will both be periodic.

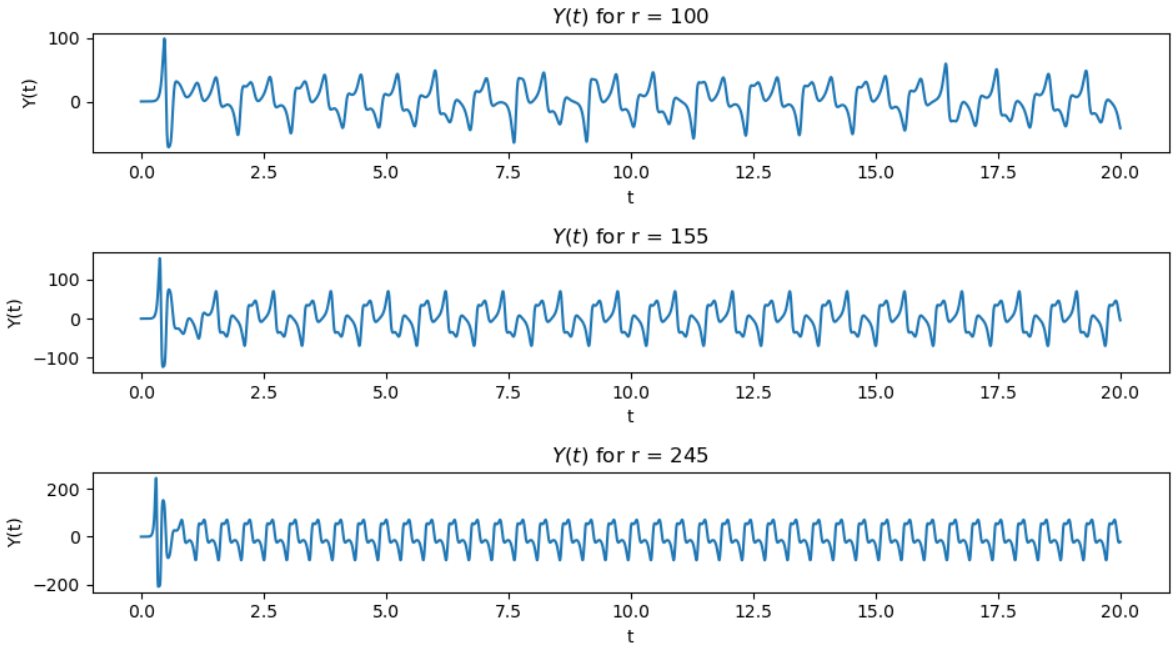

FIG. 8. Plot of $Y(t)$ for different values of r. Made with python

As we can see in figure 8, we can clearly see that $r = 155$ and $r = 245$ have periodic solutions, and for $r = 100$ there is no periodicity since each peak has a different shape as we move forward in time. We can also examine the Poincaré map and phase trajectories for one of the periodic solutions to see if we get the same result that we saw in the previous section.

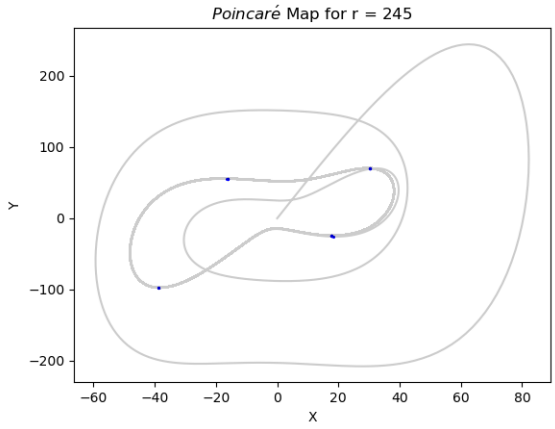

FIG. 9. Poincaré map and phase trajectory on the xy plane for r = 245. Made with python

From figure 9, we can see that there is a limit cycle, and just like with the periodic Duffing oscillator, we see a limited number of points as well! In total, we see 4 points which implies that the solutions will be oscillating about 4 values, which is the case as seen in figure 8. This means that we basically just showed and confirmed the existence of periodic solutions to the Lorenz Equations in values of r greater than 14.

V. CONCLUSION

The study of dynamical systems is a very interesting interdisciplinary subject, especially when the systems are chaotic. The applications range from weather predictions to orbital mechanics which can help engineers plan out the most efficient way to travel between planets. Although Chaos Theory really was born from differential equations, and was further popularized by Lorenz's simplification to fluid dynamic equations, there can still be chaos in discrete equations such as the logistic map. Equations like the logistic map are very easy to model and run through a computer algorithm, and the amazing thing is that it allows us to further explore structures within chaotic systems such as period doubling bifurcation. Bifurcation diagrams like the one in figure 7 take a very long time to make since we are solving the Lorenz equations thousands of times which is computationally expensive. With the logistic map, getting a high resolution bifurcation diagram is not computationally expensive, and it quickly allows us to study the behavior of chaotic dynamical systems and make general predictions for systems like the Lorenz equations. There are questions that comes from this, do all chaotic dynamical systems exhibit periodicity? If so, what does this periodicity mean in terms of real world applications? Can we extend this to the study of spatiotemporal chaos or is this only limited to temporal chaos?

VI. APPENDIX

This appendix will only contain the code that I used to make some of the plots in this document. If you want to try this out for yourself, make sure you have python installed and all the necessary libraries installed as well! The code looks much better in VSCode and also with Jupyter, but these are not neccesary and you can use any IDE you choose. The proper libraries are numpy, matplotlib, scipy, pandas, PyQt5, tqdm, and ipywidgets. The last library should automatically be installed if you have jupyter for python, the others may or may not be installed depending on where you installed python from.

A. Duffing Oscillator

```
#!/usr/bin/env python
# coding: utf-8
# Henry Pacheco Cachon
# Modeling the Duffing Oscillator with python
# Created: 12/1/2020 Updated: 12/2/2020
# In[1]:
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import interactive
import PyQt5
get_ipython().run_line_magic('matplotlib', 'qt')

# In[2]:
# Duffing Oscillator model as a function
def duffing(initialConditions, t, gamma):

    delta = 0.3
```

```
    alpha = 1
    beta = 1
    omega = 1.2

    x,y = initialConditions

    model = [y,
             -delta*y + alpha*x - beta*(x**3) + gamma * np.cos(omega * t)]

    return(model)

# In[3]:
#Setting up initial conditions and extra parameters needed to make a proper
    poincare map
period = 2*np.pi/1.2
dt = period/100
coolStep = int(period/dt)
y0 = [0,0]

# In[21]:
#Numerically solving oscillator with gamma = 0.37
t1 = np.arange(0,18000,dt)
sols1 = odeint(duffing,y0,t1, args=(0.37,))

# In[22]:
#Numerically solving oscillator with gamma = 0.50
t2 = np.arange(0,18000,dt)
sols2 = odeint(duffing,y0,t2, args=(0.50,))

# In[23]:
```

```
#Cutting solutions and time in order to make the plots look nicer
t1 = t1[33000:]
sols1 = sols1[33000:]
X1 = sols1[:,0]
V1 = sols1[:,1]

t2 = t2[33000:]
sols2 = sols2[33000:]
X2 = sols2[:,0]
V2 = sols2[:,1]

# In[25]:
#Plots of both oscillators with different gamma values
plt.subplot(211)
plt.plot(X1,V1,'k',alpha=0.45)
plt.scatter(X1[::coolStep],V1[::coolStep],10,'b')

plt.subplot(212)
plt.plot(X2,V2,'k',alpha=0.45)
plt.scatter(X2[::coolStep],V2[::coolStep],10,'b')
```

B. Lorenz Equations

```
#!/usr/bin/env python
# coding: utf-8
# In[ ]:
# Henry Pacheco Cachon
# Code to solve the Lorenz Equations and to do fun stuff with it
# Created:11/28/2020 Updated:12/2/2020

# In[1]:
```

```
get_ipython().run_cell_magic('capture', '', "import numpy as np\nimport
    matplotlib.pyplot as plt \nfrom scipy.integrate import odeint \nfrom
    mpl_toolkits.mplot3d import Axes3D\nfrom matplotlib import interactive\
    nimport PyQt5\n%matplotlib qt\n\n''' Progress bar'''\nfrom tqdm import
    tqdm_notebook as tqdm\ntqdm().pandas()")


# In[2]:
# Lorenz Model made into a function
def lorenz(initialConditions, t, r):
    y1, y2, y3 = initialConditions
    sigma = 10
    b = 8/3

    system = [-sigma * ( y1 - y2 ),
              -y1*y3 + r * y1 - y2,
              y1*y2 - b * y3]

    return(system)

# In[3]:
# Poincare map function
def poincareMap(solutions):
    # This inputs an array which is the numerical solutions to the Lorenz
    equations
    # It then runs through each value and checks if the value of z is within
    r and r+0.1
    # Then it outputs a 2d array with all the xy points for the values of z
    which satisfies that condition

```

```
32      poincTest = np.empty((2,2))
33
34      for i in range(len(solutions)):
35          X = solutions[i,0]
36          Y = solutions[i,1]
37          Z = solutions[i,2]
38
39          if np.all(Z>=r) and np.all(Z<r+.1):
40              poincTest = np.append(poincTest, [[X,Y]], axis=0)
41      poincTest = np.delete(poincTest, [0,1], 0)
42
43      return(poincTest)
44
45  # In[4]:
46  #Bifurcation diagram function
47  def bifurcation(parameter, model, y0, t, direction):
48      # This function inputs the varying parameter, a model, initial conditions
        , time, and the direction you want.
49      # NOTE the direction only has two keys, X or Y!
50      # The function runs through an array containing parameters we're
        interested in and then runs through the poincareMap function
51      # After that, it checks which direction we are interested in and it
        outputs a 1d array with those points.
52
53      Poincare = []
54
55      for r in tqdm(parameter):
56          sols = odeint(model,y0,t,args=(r,))
57
58          poincTest = poincareMap(sols)
```

```
59
60          if direction == 'X':
61              poincTest = poincTest[:,0]
62          if direction == 'Y':
63              poincTest = poincTest[:,1]
64
65          #Can change how we pick out points, right now it is picking the very
        last point in the 1d poincTest array
66          if len(poincTest) != 0:
67              Poincare.append(poincTest[len(poincTest)-1])
68          else:
69              Poincare.append(0)
70
71      return(Poincare)
72
73  # In[11]:
74  #Setting up initial conditions, t, and r
75  #Then solves using odeint function from numpy.integrate
76  y0,t0 = [0,0.001,0],0
77  t = np.linspace(0,50,10000)
78  r = 28
79  sols = odeint(lorenz,y0,t,args=(r,))
80
81  #Plotting XYZ solutions and the plane z = r
82  x,y = np.meshgrid(range(-21,21),range(-21,21))
83  fig = plt.figure()
84  ax = plt.axes(projection = '3d')
85  ax.plot3D(sols[:,0],sols[:,1],sols[:,2])
86  ax.plot_surface(x,y,x*0+y*0+r)
87  ax.view_init(azim=90, elev=15)
```

```
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

#Plotting poincaremap and XY phase trajectory
poincare = poincareMap(sols)
fig2 = plt.figure()
ax = plt.axes()

ax.plot(sols[:,0],sols[:,1],'k',alpha = 0.2)
ax.scatter(poincare[:,0],poincare[:,1],1.5,'b')

# In[7]:
#Makes bifurcation plot! Depending on how you set up R and t, this can take a
    long time!
#I put a progress bar to keep track of the progress
R = np.linspace(0,250,10000)
test = bifurcation(R,lorenz,y0,t,direction='Y')
plt.plot(R,test)
```

[1] C. Oestreicher, Dialogues in Clinical Neuroscience (2007), https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3202497/.

[2] H. Poincaré, Acta Math. **13**, 181 (1890).

[3] Z. Sun, W. Xu, X. Yang, and T. Fang, Chaos, Solitons and Fractals **27**, 705 (2006).

[4] E. N. Lorenz, Journal of the Atmospheric Sciences **20**, 130 (1963), https://journals.ametsoc.org/jas/article-pdf/20/2/130/3406061/1520-0469(1963)020_0130_dnf_2_0_co_2.pdf.