Poincare Plots, Bifurcation Diagrams, and Cobweb Plots

Amanda Rojas

Fall 2020

Abstract

This lab expands upon the concepts of previous labs, such as bifurcation and logistic maps, and adds to their concepts. We expand upon these ideas by introducing Poincare and Cobweb plots. Each of these use iterative functions in order to create chaos from simple functions. Similar to other topics described in nonlinear systems, there are applications of these concepts in nature, as they have been used to study fluctuations of the heart during ECG.

1 Introduction

1.1 Poincare Plots

The Poincare plot is a recurring plot that allows one to analyze the similarity within the function when iterated repeatedly. Poincare plots are another method of generating chaos from simple rules and it is expanded upon in the lab with in introduction of Cobweb Plots below.

1.2 Bifurcation Diagrams

Similar to the previous lab, the equation we looked at in this lab is the following:

$$x_{n+1} = rx_n(1 - x_n) \tag{1}$$

In this equation "r" is the growth rate of variable x.
"X" will be whatever it is you're measuring, you're independent variable. It could be the population, temperature, etc.
We add the term "(1-x_n)" to represent environmental constraints that restrict the variable x from exponentially increasing.
The variable "n" is obviously the amount of iterations.

1.3 Cobweb Plots

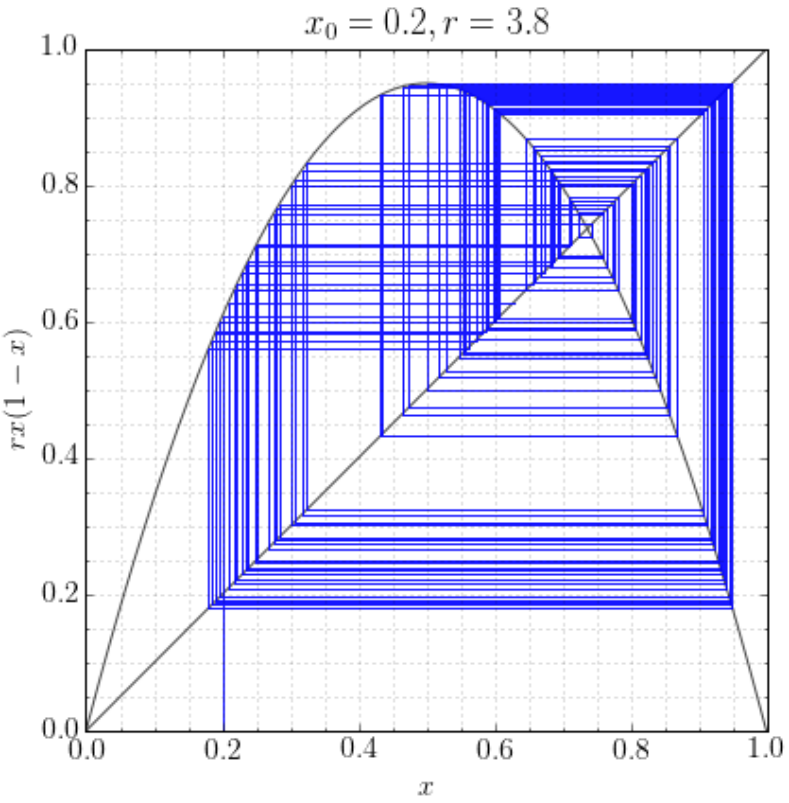
Cobweb plots are another method of generating chaos from simple rules and very similar to the bifurcation we saw in last week's lab report, there is order up until a certain point, than chaos, then periods of order again.

2 Theory– Cobweb Plots

In order to plot a cobweb plot, a parabolic function is required. In this lab we used the following equation:

$$y = rx(1 - x) \tag{2}$$

First you plot this equation.
Next you plot the function y=x
Then you pick any random point under the parabola and draw a line up to the parabola.
Then draw a line from the current point on the parabola to the same y coordinate on the y=x line.
Then draw a line from the current point on the linear function to the same x coordinate on the parabola. Repeat.
Eventually your plot will erupt into chaos, with periods of stability.



3 Procedures

Logistic Map Part 2: Poincare Plots, Bifurcation Diagrams, and Cobweb Plots

```
#imports python number/plotting libraries respectively
import numpy as np
import matplotlib.pyplot as plt
```

```
#defines the logisitc function
def logistic(r, x):
  return r * x * (1-x)
```

```
#defines n, and generates values of n between 2.5 and 4.0
n = 100
r = np.linspace(2.5, 4.0, n)
```

```
#defines x
x= 1e-5 * np.ones(n)
```

```
#defines number of iterations
iterations = 150
```

```
x = logistic(r, x)
```

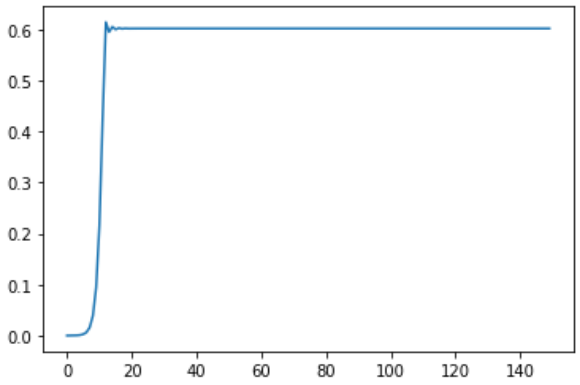
```
#creates a matrix from the amount of iterations and n
X = np.zeros((iterations, n))
```

```
#assigns the values of the logistic function, that being 1 to the iterations
for i in range(1, iterations):
  x = logistic(r,x)
  X[i, :] = x
```

```
#plots the first column
plt.plot(X[:,[1]])
```

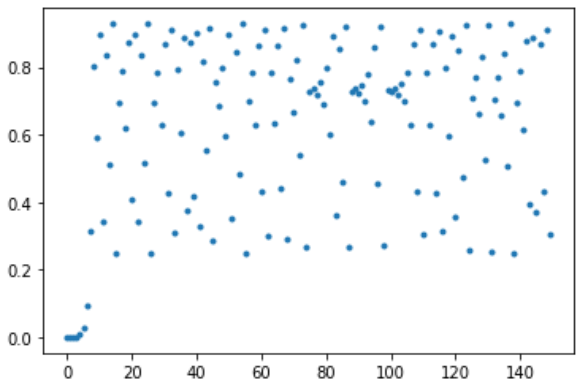
```
#plots the first column
plt.plot(X[:,[1]])
```

```
[<matplotlib.lines.Line2D at 0x7f466646a160>]
```



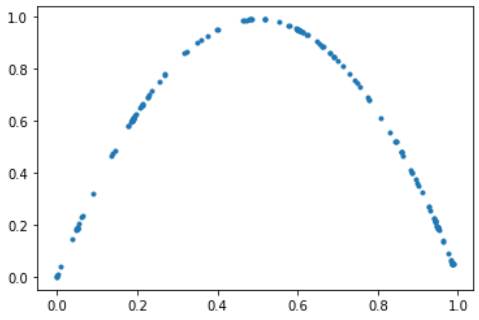

```
#plots the 81st column
plt.plot(X[:,80], '.')
```

```
[<matplotlib.lines.Line2D at 0x7f4665fa4978>]
```



```
#plots both matricies (X[0:-1,n1] and X[1:,n1]) against one another
plt.plot(X[0:-1,n1], X[1:,n1], '.')
```

```
[<matplotlib.lines.Line2D at 0x7f4665f26c50>]
```



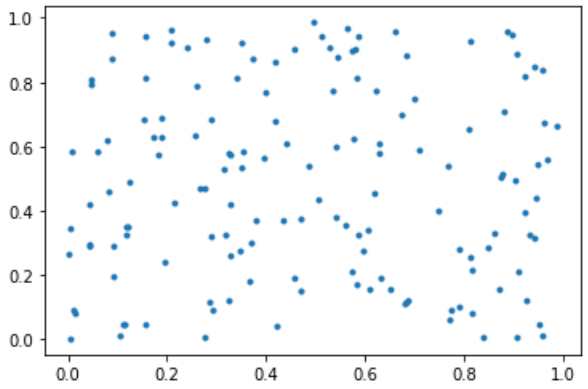
```
#identifying the shape
X.shape
```

```
(150, 100)
```

```
#doing the matrix with random values
X1 = np.random.random(X.shape)
```

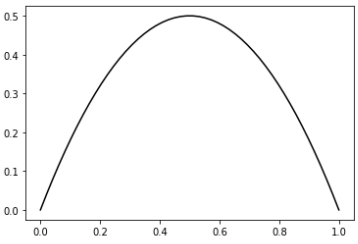
```
#plots the new matrix with random values
plt.plot(X1[0:-1,n1], X1[1:,n1],'.')
```

```
[<matplotlib.lines.Line2D at 0x7f4665e8af60>]
```



```
#defines the intervals between 0 and 1, the plot axis, and subsequently plots the figure.
x = np.linspace(0, 1)
fig, ax = plt.subplots(1, 1)
ax.plot(x, logistic(2, x), 'k')
```

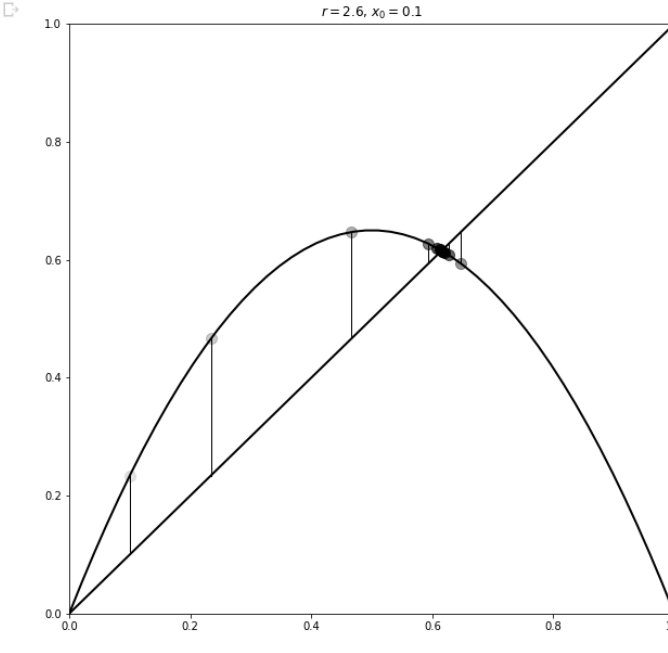
```
[<matplotlib.lines.Line2D at 0x7f4665bc6940>]
```



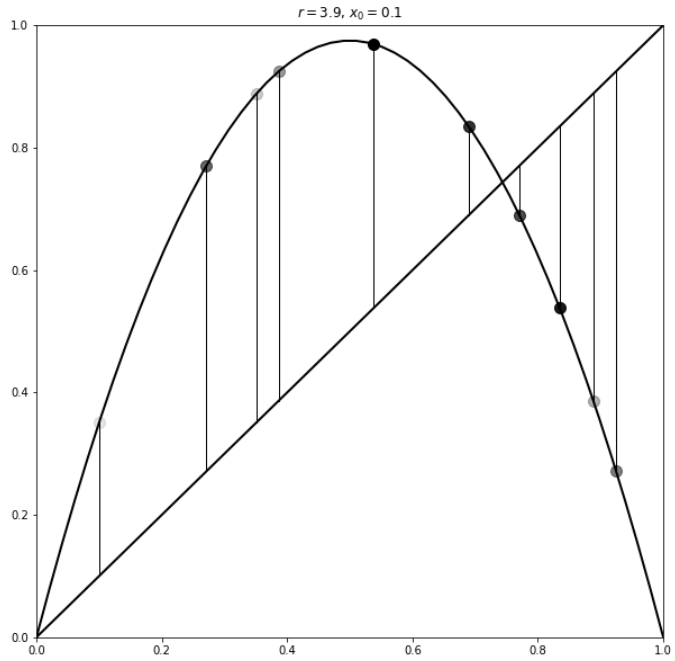
```
#defines the intervals between 0 and 1, the plot axis, and subsequently plots the figure.
def plot_system(r, x0, n):
  t = np.linspace(0, 1)

  fig, ax = plt.subplots(1, 1, figsize=(10, 10))
  ax.plot(t, logistic(r, t), 'k', lw= 2)
  ax.plot([0, 1], [0,1], 'k', lw=2)

  x= x0

  #iterates the function
  for i in range(n):
    y = logistic(r, x)

    ax.plot([x, x], [x, y], 'k', lw= 1)
    ax.plot([x, x], [y, y], 'k', lw= 1)
    ax.plot([x], [y], 'ok', ms= 10, alpha=(i + 1) / n)

    x = y
```

```
  ax.set_xlim(0, 1)
  ax.set_ylim(0, 1)
  ax.set_title(f'$r={r:.1f}, \, x_0={x0:.1f}$')
```

```
#plots
plot_system(2.6, .1, 10)
```



```
[29] plot_system(3.9, .1, 10)
```



4 Conclusions

This lab expands upon the ideas of the previous labs and demonstrates the diversity of complex systems. Poincare and cobweb plots expands upon the idea that complex behaviors can arise from simple executions. This expands upon the idea that perhaps the seemingly random outcomes that emerge from complex systems may in fact be simple phenomena interacting repeatedly.