Exploring the Logistics Map through Python

Zubair Ali

Abstract – Discrete-time systems are often used to model problems where the state-space is too large to be continuous over time, hence it is known to be operated on computer simulations. In this context, we consider the logistic map, a one dimensional discrete-time map that exhibits complex chaotic behavior, with the interest of modeling the logistic map on the programming language, Python, to illustrate period-doubling bifurcation route to chaos over a fixed period of time.

1 Introduction

The logistic map effectively models population dynamics by accounting for the limited availability of resources, balancing reproductive growth against environmental constraints. It demonstrates the interaction between reproduction rates and resource availability, where excessive growth is naturally curbed by environmental limits. The logistic map equation we will be modeling is: $x_{n+1} = r \cdot x_n \cdot (1 - x_n)$with variables and parameters defined as:

- x_n : Represents the state of the system at time n
- **r**: The rate parameter, controlling the system's growth or decline.

As the parameter r varies, the logistic map showcases a transition from predictable behavior to chaos, serving as a classic example of how simple deterministic rules can lead to complex and unpredictable patterns. The system's evolution is governed by iterative applications of a specific function or set of functions.

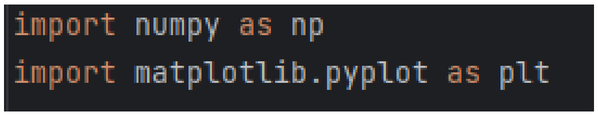
2 Code

2.1 Python Libraries

The code we are writing will be done through Python. Although Python has various libraries we will only be implementing NumPy and Matplotlib.

- NumPy, short for Numerical Python, is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays. In our logistic map implementation, NumPy's array structures allow us to efficiently store and manipulate large sets of numerical data, essential for calculating the iterations over a range of r values.
- Matplotlib is a plotting library in Python that enables us to create a wide range of static, animated, and interactive visualizations. It's known for its flexibility and ease of use in generating graphs and charts. For the logistic map, Matplotlib helps us in visually representing the data, allowing us to plot the bifurcation diagram and observe the transition from order to chaos as r varies.

After these libraries are installed, they have to be imported in order to be able to run the code.

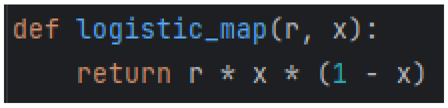


```
import numpy as np
import matplotlib.pyplot as plt
```

2.2 Logistic Map Equation

After all our libraries have been imported, we move on to writing the code for the logistics map, which will be labeled under logistic_map. The logistic_map function is the heart of our Python implementation. It defines the logistic map equation, $\mathbf{x_{n+1} = r \cdot x_n \cdot (1 - x_n)}$, which we use to simulate the behavior of the logistic map over time. We define the parameters as,

- **r** (float): This parameter represents the rate of growth or reproduction. It's a key factor that influences how the population (or the value of x) changes with each iteration.
- **x** (float): This is the current value or state of our system, representing, for example, the current population ratio.



```python
def logistic_map(r, x):
    return r * x * (1 - x)
```

Each time the function is called, it calculates the next value in the sequence based on the current value and the growth rate. This iterative process is what allows us to observe the behaviors of the logistic map, from stable states to chaotic fluctuations.

2.3 Parameters

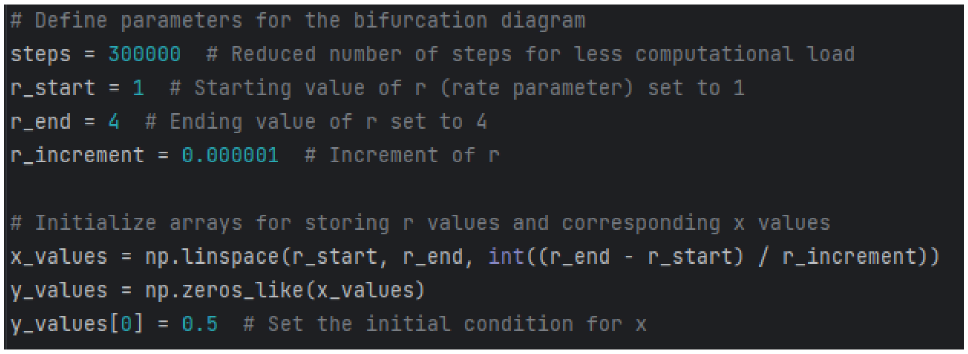
Now that we have the logistics map function defined, we move on to defining the parameters for the bifurcation diagram.

- steps: Number of iterations to perform, set to 300,000 for detailed analysis with manageable computational load.

- r_start and r_end: Define the range of the rate parameter r, starting from 1 and ending at 4, focusing on a specific segment of the logistic map.

- r_increment: The step size for incrementing r, chosen to be small (0.000001) for high resolution in the diagram.

To initialize the arrays for the r and x values, we will be utilizing the built in library, NumPy's linspace function to create an array of r values from r_start to r_end. This array represents the different growth rates we will explore in the logistic map. Then an array of y_values is initialized to store the corresponding x values for each r. It's set to the same size as x_values and starts with an initial condition of 0.5, representing our starting point for the population ratio or the state of the system.



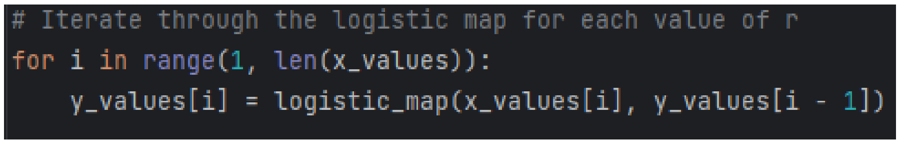
```
# Define parameters for the bifurcation diagram
steps = 300000  # Reduced number of steps for less computational load
r_start = 1  # Starting value of r (rate parameter) set to 1
r_end = 4  # Ending value of r set to 4
r_increment = 0.000001  # Increment of r

# Initialize arrays for storing r values and corresponding x values
x_values = np.linspace(r_start, r_end, int((r_end - r_start) / r_increment))
y_values = np.zeros_like(x_values)
y_values[0] = 0.5  # Set the initial condition for x
```

2.4 Simulating the Logistic Map

Following the initialization for the arrays, is simulating the logistic map. This is done by iterations in which we simulate the behavior of the logistic map for each value of the growth rate r. Using a loop, we iterate through each r value stored in x_values. In each iteration, the logistic_map function is called to calculate the next x value, given the current x value and the r value. This iterative process allows us to explore how the system evolves over time for different growth rates.

We build the bifurcation diagram by iterating through each r, the corresponding x values are stored in y_values. This array accumulates the final state of the system after numerous iterations for each r, which is crucial for plotting the bifurcation diagram. The logic y_values[i] = logistic_map(x_values[i], y_values[i - 1]) ensures that each new x value is calculated based on the most recent x value, maintaining the continuity and dependency of the system's state over time.

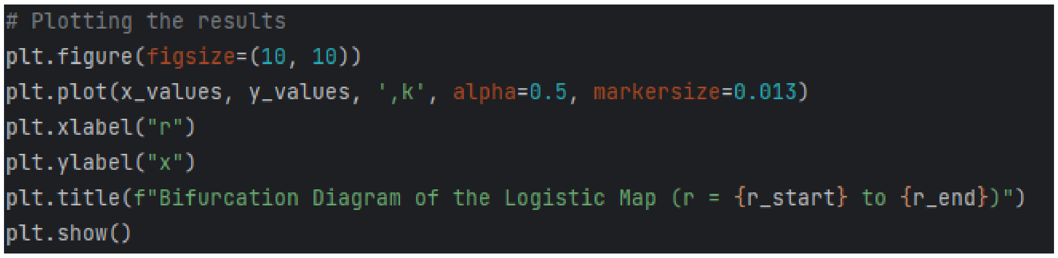


```
# Iterate through the logistic map for each value of r
for i in range(1, len(x_values)):
    y_values[i] = logistic_map(x_values[i], y_values[i - 1])
```

2.5 Plotting Bifurcation Diagram

Lastly, we make use of the library Matplotlib to create the plot for the bifurcation diagram. We begin by creating a figure with plt.figure(figsize=(10, 10)), specifying the size of the plot. The plt.plot function is then used to draw the diagram. Here, x_values (representing different r values) and y_values (corresponding system states) are plotted.

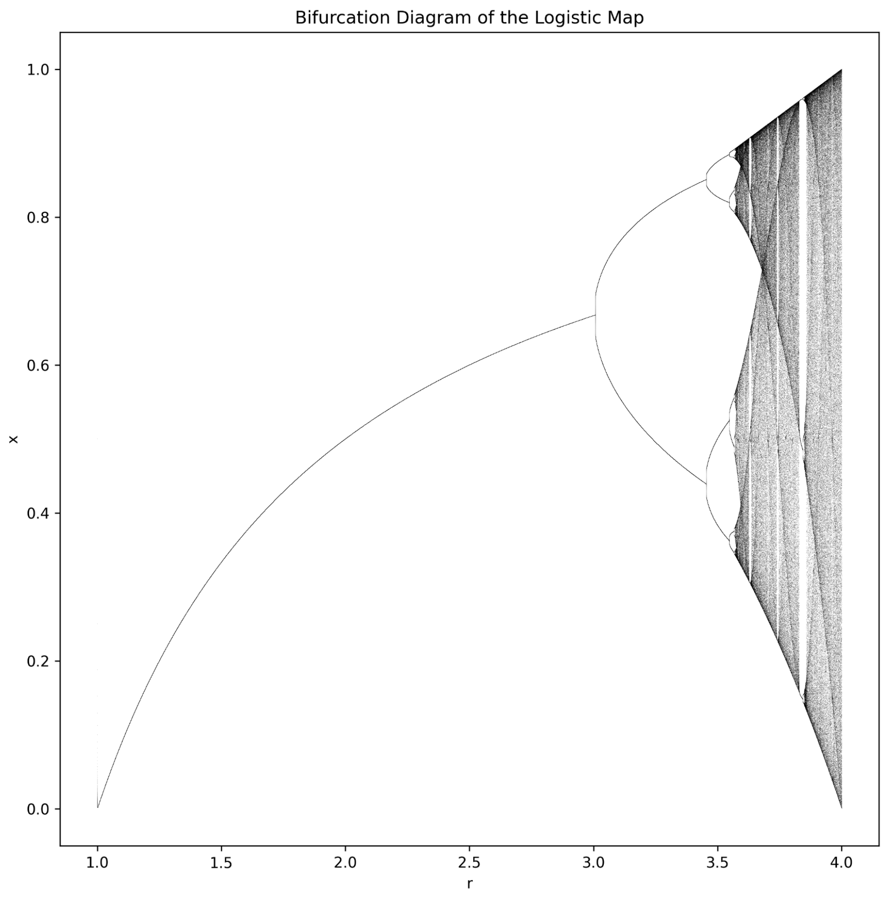
We customize the plot by the arguments ',k', alpha=0.5, and markersize=0.013 in plt.plot to define the appearance of the plot points. ',k' sets the color to black, alpha adjusts the transparency, and markersize controls the size of each point. Labels for the x-axis and y-axis are set with plt.xlabel("r") and plt.ylabel("x"). The title of the plot is set with plt.title.



```
# Plotting the results
plt.figure(figsize=(10, 10))
plt.plot(x_values, y_values, ',k', alpha=0.5, markersize=0.013)
plt.xlabel("r")
plt.ylabel("x")
plt.title(f"Bifurcation Diagram of the Logistic Map (r = {r_start} to {r_end})")
plt.show()
```

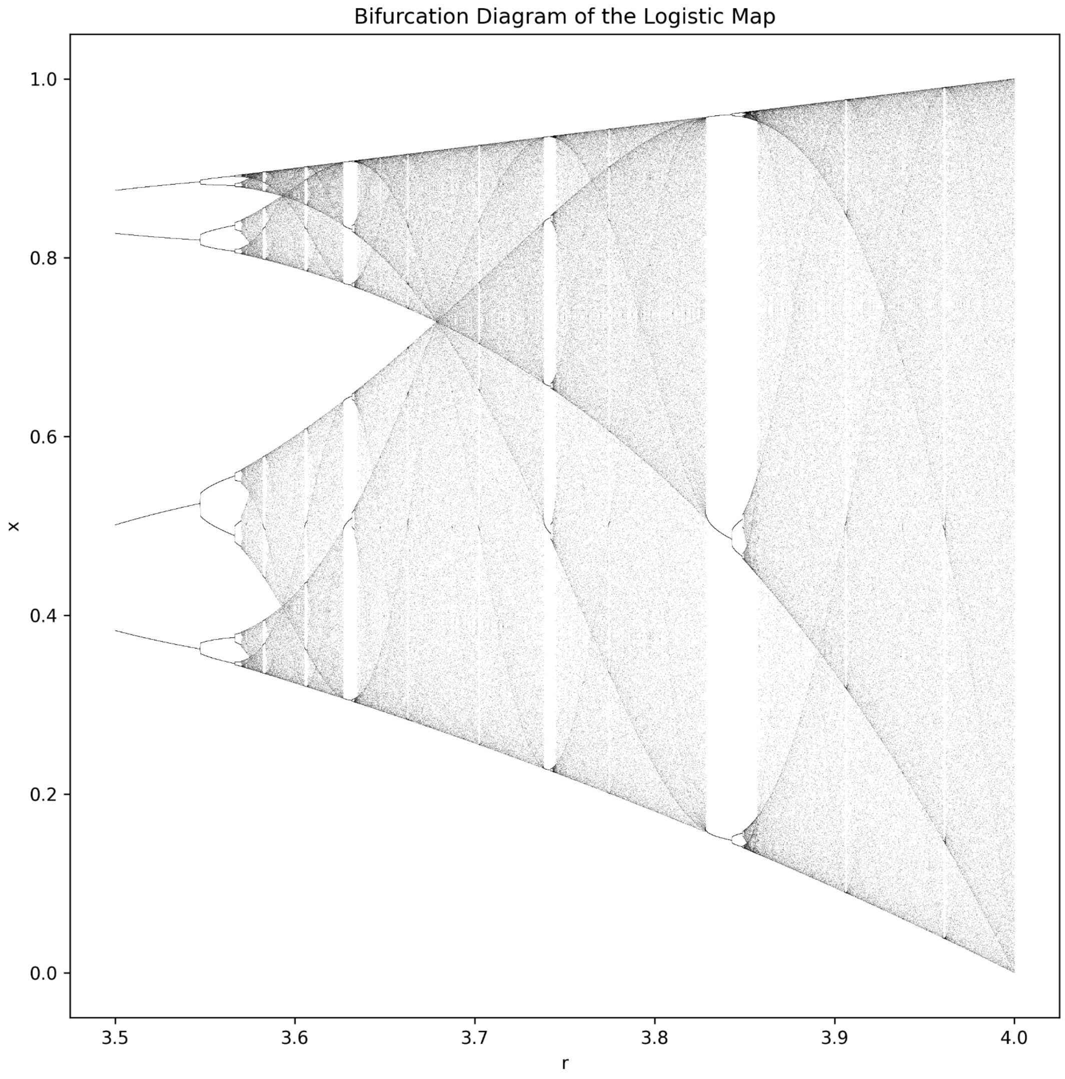
3 Discussion

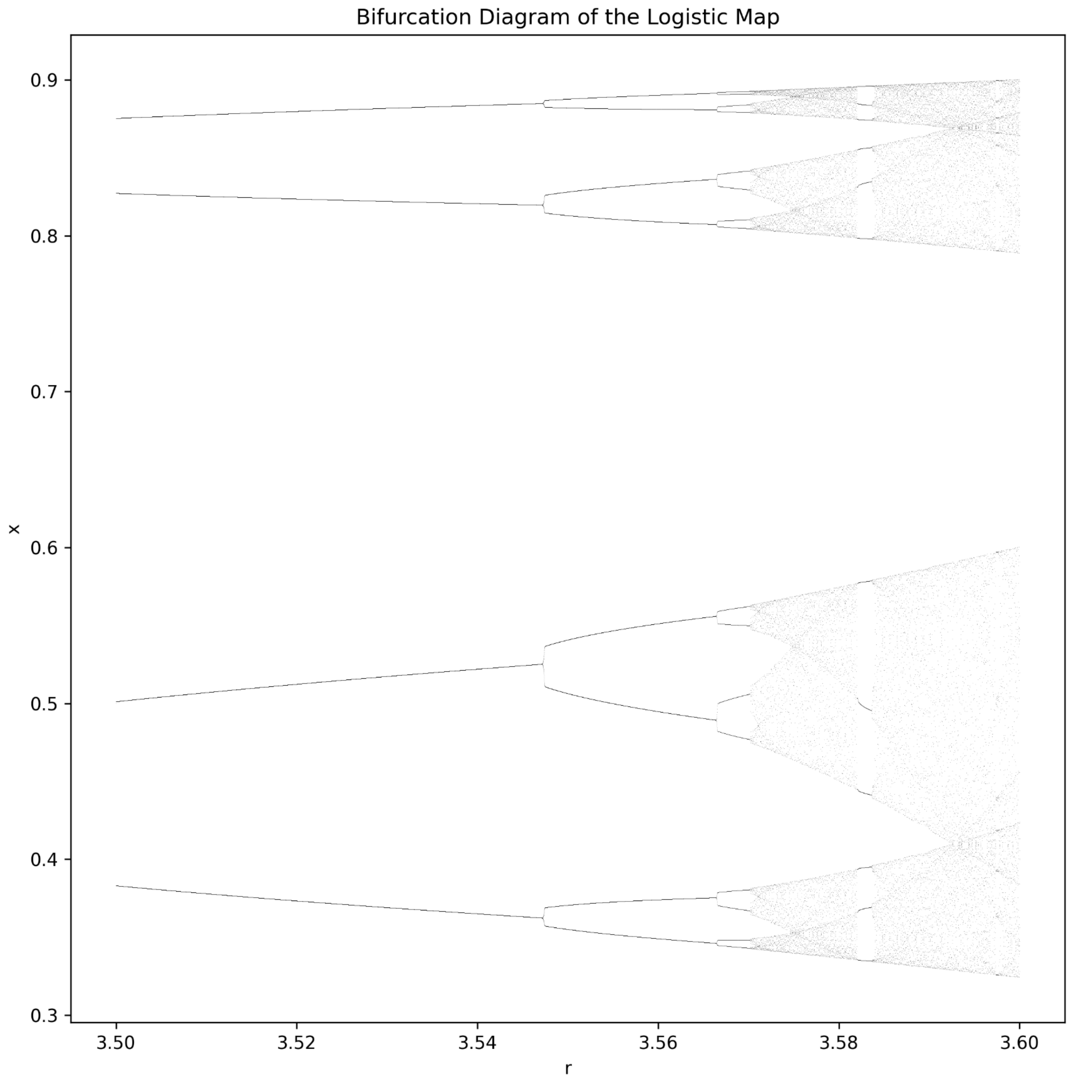
After completing the code, plt.show() is called, in which the completed bifurcation diagram is successfully displayed, showcasing the logistic map within the specified r interval, [0,4].



4 Further Work

Since we were able to create the model for the logistics map between the interval [0,4], we further explored the chaos by "zooming in," on the intervals [3.5,4] and [3.5,3.6] which allowed us to demonstrate the period-doubling bifurcation.


Bifurcation Diagram of the Logistic Map
1.0
0.8
0.6
0.4
0.2
0.0
x
3.5
3.6
3.7
3.8
3.9
4.0
r



We were interested in running the plot in real time but had a hard time writing the code. The closest we were able to get the animation working resulted in parts of the bifurcation missing. We were also interested in color coding the plot by each bifurcation for the bifurcations to stand out, however we ran out of time to complete that part of the code.

5 References

1. “The Logistic Map.” Complexity Explorables, 30 Apr. 2018, www.complexity-explorables.org/flongs/logistic/.

2. Logistic Map.” Wikipedia, Wikimedia Foundation, 8 Nov. 2023, en.wikipedia.org/wiki/Logistic_map.

3. “What Is Discrete System?: Autoblocks Glossary.” Autoblocks, www.autoblocks.ai/glossary/discrete-system. Accessed 11 Dec. 2023.