# AMATH Problem Set 3 Question 3

Leo Wei

August 2023

## 1 Consider the logistic difference equation

$$x_{n+1} = ax_n(1 - x_n)$$

### a. we constrain $x_i$ to the interval $[0, 1]$. Determine $a_{min}$ and $a_{max}$ for which the difference equation is well-defined (i.e. $x_i \in [0, 1]$ for all $i \geq 0$).

$a_{min}$ is achieved when we want the expression to be non-negative. From the equation, we can see that as long as $x_n$ is in range of $(0, 1)$, the value of $a$ does not impact the equation, so $a_{min} = 0$.
To find $a_{max}$, we first set $x_{n+1} \to f(x) = x(1 - x)$ to simplify the calculations. The maximum of $f(x)$ is when $f'(x) = 0$

$$f'(x) = 1 - 2x = 0; x = \frac{1}{2}$$

Maximum of $f(x)$ is at $x = \frac{1}{2}$. Thus, plugging in $x = \frac{1}{2}$, we obtain

$$f(\frac{1}{2}) = \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{4}$$

We want $x_{n+1}$ to be within the interval $x_i \in [0, 1]$. So we set the interval

$$0 \leq x_n(1 - x_n) \leq \frac{1}{4}\ for x_i \in [0, 1]$$

To find $a_{max}$, we multiply it into the interval (To obtain the original equation)

$$0 \leq ax_n(1 - x_n) \leq \frac{a}{4} \to 0 \leq x_{n+1} \leq \frac{a}{4}$$

From this we can infer that $\frac{a}{4} \leq 1$ since we are in the interval $x_i \in [0, 1]$. We then solve for $a$

$$a \leq 4$$

Since we already have $a_{min}$, this lets us know that $a_{max} = 4$.
Therefore, we can conclude that $0 \leq a \leq 4$

**b. For a difference equation $x_{n+1} = f(x_n)$, an equilibrium point $x^*$ satisfies $x^* = f(x^*)$. Determine all equilibria of the logistic difference equation.**

To find all the equilibria, we solve for

$$x^* = ax^*(1 - x^*)$$
$$x^* - ax^* + ax^*2 = 0 \to x^*(1 - a + ax^*) = 0$$
$$1 - a + ax^* = 0$$
$$x_1^* = 0,\ x_2^* = \frac{a-1}{a}$$

**c. For a difference equation $x_{n+1} = f(x_n)$, an equilibrium point $x^*$ is asymptotically stable if $|f'(x^*)| < 1$. Furthermore, if $0 < f'(x^*) < 1$, the system approaches the equilibrium monotonically; if $-1 < f'(x^*) < 0$, the system approaches the equilibrium oscillatory. Analyze in detail the stability of each equilibrium for our difference equation.**

From part b., we find that the two equilibrias are $x_1^* = 0$ and $x_2^* = \frac{a-1}{a}$. From the given information above, we can test the equilibria's stability by taking the derivative at that point. We first solve for the derivative with respect to $x_n$

$$\frac{d}{dx_n} x_{n+1} = \frac{d}{dx_n}(ax_n(1 - x_n))$$
$$x'_{n+1} = \frac{d}{dx_n} ax_n - \frac{d}{dx_n} ax_n^2 = a - 2ax_n$$

We now plug in the equilibria.
For $x_1^* = 0$

$$f'(0) = a - 0 = a$$

For $x_2^* = \frac{a-1}{a}$

$$f'(\frac{a-1}{a}) = a - 2a(\frac{a-1}{a}) = a - 2a + 2 = -a + 2 = 2 - a$$

The stability of each equilibria depends on the value of $a$.
For $x_1^*$:

1. For $a = 0$, the equilibria is stable since $|f'(0) < 1|$
2. For $a = 1$, the equilibria is marginally stable since $|f'(0) = 1|$
3. For $a = 2, 3, 4$, the equilibria is be unstable since $|f'(0) > 1|$

For $x_2^*$:

1. For $a = 2$, the equilibria is stable since $|f'(0) < 1|$
2. For $a = 1, 3$, the equilibria is marginally stable since $|f'(0) = 1|$
3. For $a = 0, 4$, the equilibria is be unstable since $|f'(0) > 1|$

**d. Find the value $a_2 \in [a_{min}, a_{max}]$ after which 2-period oscillations begin. Plot the position of 2 period oscillations for $a \in [a_2, a_{max}]$ . Hint: $f$ exhibits $n$ period oscillations at $x$ if $x = f_n(x)$ and $x \neq f_m(x)$ for $0 < m < n$ ($f_n$ is $f$ composed with itself $n$ times $f * f * \cdots * f$).**

For this question, we are looking for $a_2$, which is $a$ after 2 period oscillations. We will use the following steps:

1. Choose a range $a$ values to examine the behavior of. Through countless testing, I choose $a$ values from 3 to 4

2. Iterating the logistic map using the provided equation for each $a$ value. We need a large enough iteration to find the system's long term behavior

3. We record the last 2 iterations of each $a$ value, since these represent the stable behavior of that system. If we detect 2-period oscillations, the last 2 values will be different but it will repeat every 2 iterations.

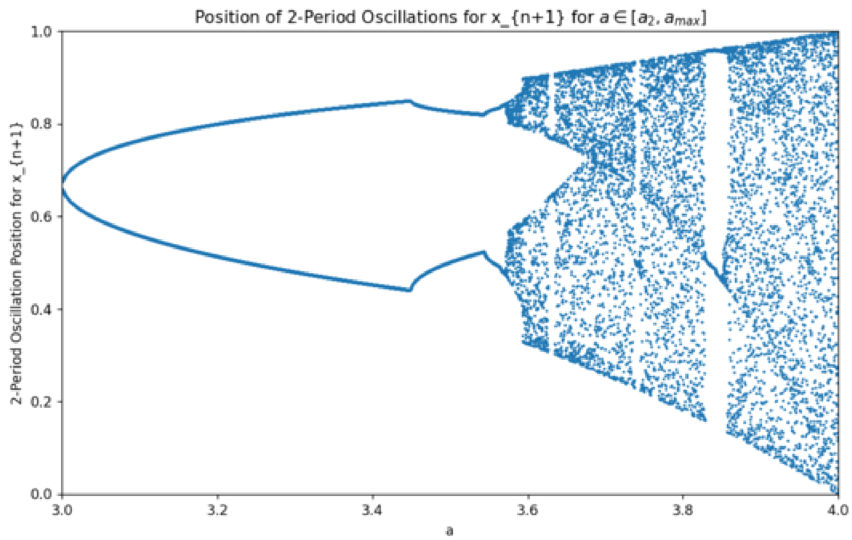
4. As mentioned above, the last 2 values have to be different to indicate potential 2-period oscillations, so we have to detect if the last 2 values are different

5. Use python to find $a_2$

Here is the code to solve:

```python
# Importing necessary libraries
import numpy as np

# Function to iterate the logistic map and return the last few iterations
def iterate_logistic_map(a_value, initial_value = 0.5, total_iterations = 1000, last_iterations = 2):
    x_value = initial_value
    last_values = []
    for i in range(total_iterations):
        x_next = a_value * x_value * (1 - x_value)
        if i >= total_iterations - last_iterations:
            last_values.append(x_next)
        x_value = x_next
    return last_values

# Parameters
a_values = np.linspace(3, 4, 10000)  # Range of a values to analyze
last_iterations_to_record = 2  # Number of last iterations to record

# Iterate through values of a and record the last few iterations
a_2_found = False
for a_value in a_values:
    last_values = iterate_logistic_map(a_value, last_iterations = last_iterations_to_record)
    # Check for 2-period oscillations (last two values are different but repeat after two iterations)
    if abs(last_values[0] - last_values[1]) > 1e-5 and a_2_found == False:
        a_2 = a_value
        a_2_found = True
        break

print(a_2)
```

From the code, we find that $a_2 = 3$ Here is the plot:




Here is the code for the plot:

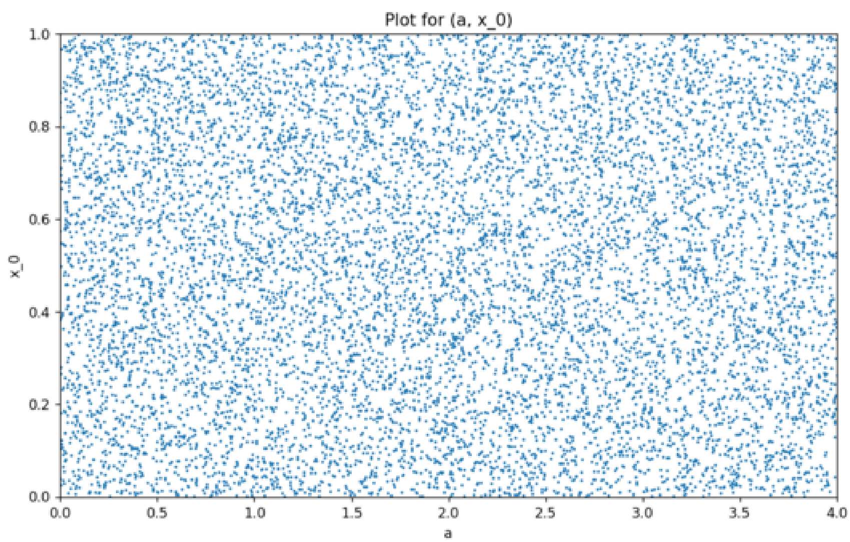
```python
# Importing necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# Function to iterate the logistic map and return the last few iterations
def iterate_logistic_map(a_value, initial_value = 0.5, total_iterations = 1000, last_iterations = 2):
    x_value = initial_value
    last_values = []
    for i in range(total_iterations):
        x_next = a_value * x_value * (1 - x_value)
        if i >= total_iterations - last_iterations:
            last_values.append(x_next)
        x_value = x_next
    return last_values

# Parameters for identifying 2-period oscillations
a_values_2_period = np.linspace(3, 4, 10000)
last_iterations_to_record = 2

# Lists to store a values and corresponding 2-period oscillation positions, corrected
a_2_period_corrected = []
x_2_period_corrected = []

# Iterate through values of a and identify 2-period oscillations
for a_value in a_values_2_period:
    last_values = iterate_logistic_map(a_value, last_iterations=last_iterations_to_record)
    # Check for 2-period oscillations (last two values are different but repeat after two iterations)
    if abs(last_values[0] - last_values[1]) > 1e-5:
        a_2_period_corrected.extend([a_value] * last_iterations_to_record)
        x_2_period_corrected.extend(last_values)

# Plotting the positions of 2-period oscillations, corrected
plt.figure(figsize=(10, 6))
plt.scatter(a_2_period_corrected, x_2_period_corrected, s = 1)
plt.title('Position of 2-Period Oscillations for x_{n+1} for $a \in [a_2, a_{max}]$')
plt.xlabel('a')
plt.ylabel('2-Period Oscillation Position for x_{n+1}')
```

```python
# Plotting the positions of 2-period oscillations, corrected
plt.figure(figsize=(10, 6))
plt.scatter(a_2_period_corrected, x_2_period_corrected, s = 1)
plt.title('Position of 2-Period Oscillations for x_{n+1} for $a \in [a_2, a_{max}]$')
plt.xlabel('a')
plt.ylabel('2-Period Oscillation Position for x_{n+1}')
plt.xlim(3, 4)
plt.ylim(0, 1)
plt.show()
```

**e. Randomly generate $10,000$ randomly distributed pairs of $a$ and $x_0$, that is, $10,000$ points on $[a_{min}, a_{max}] \times [0, 1]$ Plot all the pairs. Each part (e1) to (e6) should be a separate plot.**

$(e1)(a, x0)$

Plot:

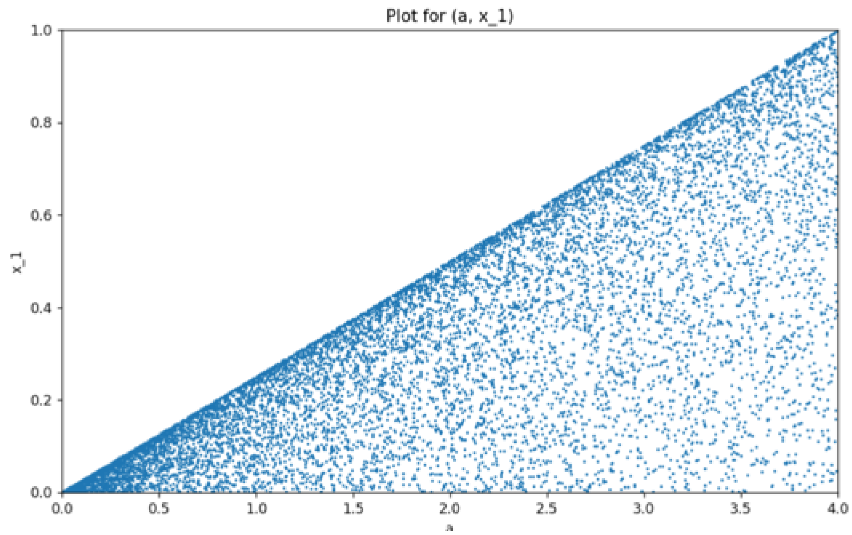



Code:

```python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Plotting the pairs (a, x0) for (e1)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, random_x0_values, s = 1)
plt.title('Plot for (a, x_0)')
plt.xlabel('a')
plt.ylabel('x_0')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

$(e2)(a, x_1)$

Plot:

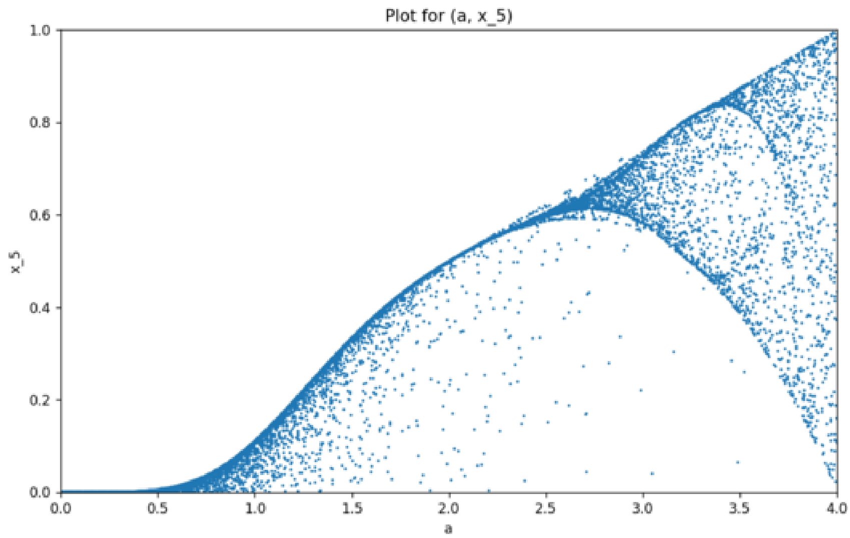



Code:

```python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Calculating x1 values after one iteration
x1_values = [logistic_map(a_value, x0_value, 1) for a_value, x0_value in zip(random_a_values, random_x0_values)]

# Plotting the pairs (a, x1) for (e2)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, x1_values, s = 1)
plt.title('Plot for (a, x_1)')
plt.xlabel('a')
plt.ylabel('x_1')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

$(e3)(a, x_5)$

Plot:

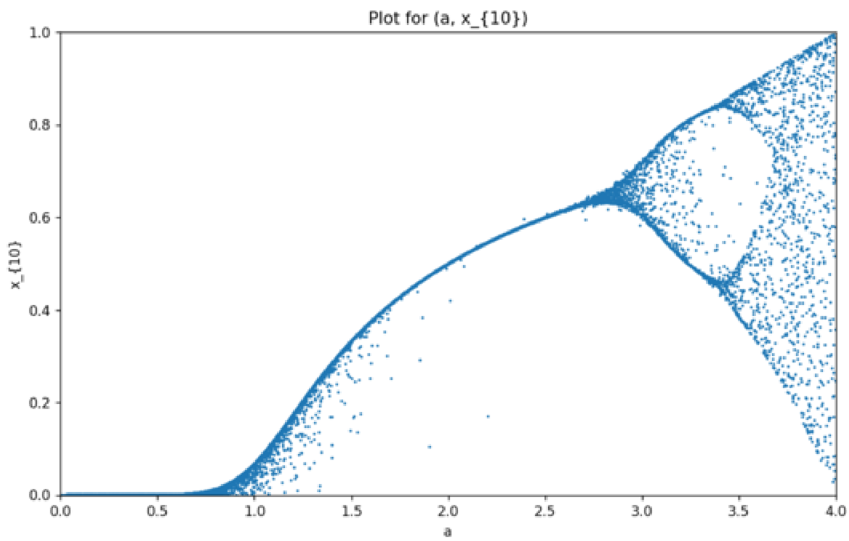



Code:

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Calculating x5 values after five iterations
x5_values = [logistic_map(a_value, x0_value, 5) for a_value, x0_value in zip(random_a_values, random_x0_values)]

# Plotting the pairs (a, x5) for (e3)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, x5_values, s = 1)
plt.title('Plot for (a, x_5)')
plt.xlabel('a')
plt.ylabel('x_5')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

$(e4)(a, x_{10})$

Plot:

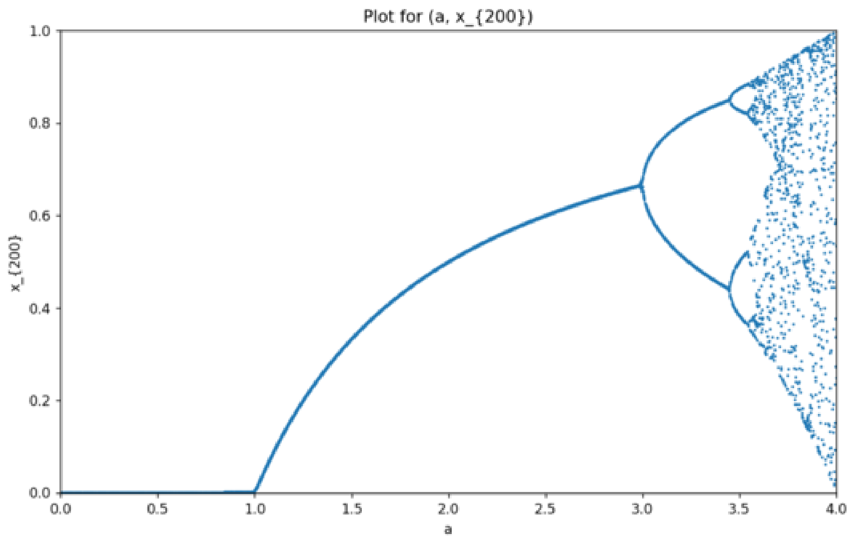



Code:

```python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Calculating x10 values after ten iterations
x10_values = [logistic_map(a_value, x0_value, 10) for a_value, x0_value in zip(random_a_values, random_x0_values)]

# Plotting the pairs (a, x10) for (e4)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, x10_values, s = 1)
plt.title('Plot for (a, x_{10})')
plt.xlabel('a')
plt.ylabel('x_{10}')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

$(e5)(a, x_{200})$

Plot:

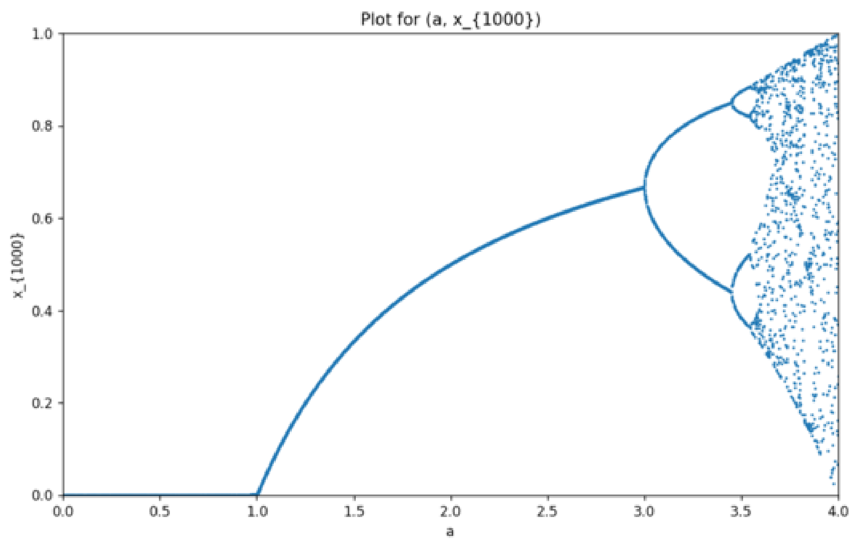



Code:

```python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Calculating x200 values after 200 iterations
x200_values = [logistic_map(a_value, x0_value, 200) for a_value, x0_value in zip(random_a_values, random_x0_values)]

# Plotting the pairs (a, x200) for (e5)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, x200_values, s = 1)
plt.title('Plot for (a, x_{200})')
plt.xlabel('a')
plt.ylabel('x_{200}')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

$(e6)(a, x_{1000})$

Plot:




Code:

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Generating 10000 randomly distributed pairs of a and x0
np.random.seed(42) # Ensuring reproducibility
random_a_values = np.random.uniform(0, 4, 10000)
random_x0_values = np.random.uniform(0, 1, 10000)

# Calculating x1000 values after 1000 iterations
x1000_values = [logistic_map(a_value, x0_value, 1000) for a_value, x0_value in zip(random_a_values, random_x0_values)]

# Plotting the pairs (a, x1000) for (e6)
plt.figure(figsize = (10, 6))
plt.scatter(random_a_values, x1000_values, s = 1)
plt.title('Plot for (a, x_{1000})')
plt.xlabel('a')
plt.ylabel('x_{1000}')
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.show()
```

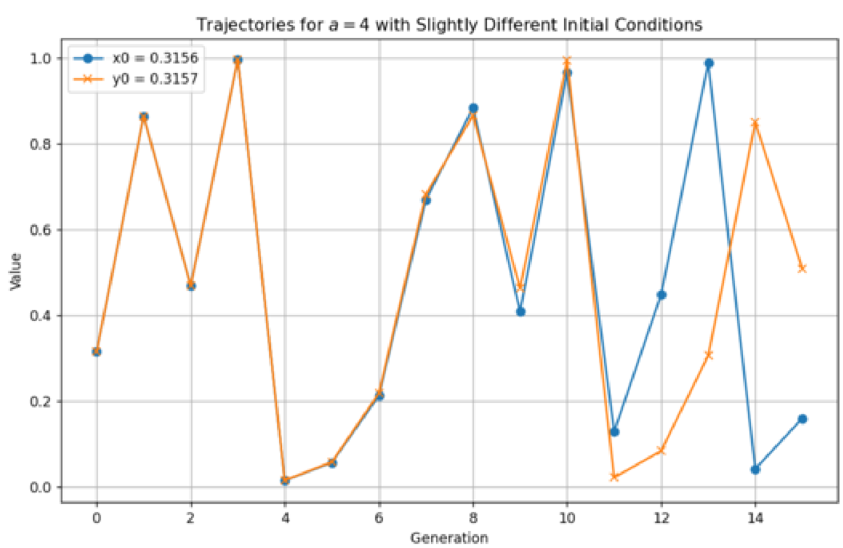
## f. describe what you see in plots from (e).

The plots illustrate the behavior of the logistic equation at different stages of iteration.
For e1, it is exhibiting random points as it setting up the stage for the following plots.

For e2, we start to see the initial effects of the logistic plot.
For e3, patterns began to emerge, where in some regions the patterns stabilize while in other they bifurcate.
For e4, The respective regions began to become more obvious. It is also here that we begin to see chaotic regions.
For e5, the system has settled into its long term behavior, it shows fixed points, periodic oscillations, bifurcations, and chaotic regions.
For e6, the pattern has become well-established, with pronounced periodic and chaotic regions.

## g. A chaotic system is a deterministic, non-linear system that appears to behave randomly. In chaotic systems small perturbations in the initial conditions have tremendous effects and lead to very different outcomes. The logistic map with $a = 4$ is a simple and most illuminating example. Describe what you see for $a$ß4 in the above simulations (5 points). Set $a = 4$ and start with two points, $x_0 = 0.3156$ and $y_0 = 0.3157$ and plot their trajectories for 15 generations.

From above, we can see that despite the system being modeled after a deterministic equation, the system is random and unpredictable. Slight variations to the initial conditions can lead to chaotic outcomes, and at $a = 4$, the system never settles into a stable cycle.
Plot:




Code:

question_3_g.py > ...

```python
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Defining the logistic map function
def logistic_map(a_value, x_value, iterations):
    for _ in range(iterations):
        x_value = a_value * x_value * (1 - x_value)
    return x_value

# Function to generate trajectory
def generate_trajectory(a, initial_value, generations):
    trajectory = [initial_value]
    for _ in range(generations):
        next_value = logistic_map(a, trajectory[-1], 1)
        trajectory.append(next_value)
    return trajectory

# Parameters for chaotic behavior
a_chaos = 4
x0_chaos = 0.3156
y0_chaos = 0.3157
generations = 15

# Generating trajectories for x0 and y0
trajectory_x0 = generate_trajectory(a_chaos, x0_chaos, generations)
trajectory_y0 = generate_trajectory(a_chaos, y0_chaos, generations)

# Plotting the trajectories
plt.figure(figsize = (10, 6))
plt.plot(trajectory_x0, label = f"x0 = {x0_chaos}", marker='o')
plt.plot(trajectory_y0, label = f"y0 = {y0_chaos}", marker='x')
plt.title('Trajectories for $a = 4$ with Slightly Different Initial Conditions')
plt.xlabel('Generation')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
```


```python
# Plotting the trajectories
plt.figure(figsize = (10, 6))
plt.plot(trajectory_x0, label = f"x0 = {x0_chaos}", marker='o')
plt.plot(trajectory_y0, label = f"y0 = {y0_chaos}", marker='x')
plt.title('Trajectories for $a = 4$ with Slightly Different Initial Conditions')
plt.xlabel('Generation')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
```


```python
plt.show()
```