Problem Set #2 —— Bifurcations

University of Colorado - Boulder
CSCI4446 — Chaotic Dynamics

JACK MATHEWS
john.mathewsiii@colorado.edu

January 27, 2026

1 Logistic Map Bifurcation Diagram

Figure 1 contains a visualization of the bifurcations of the logistic map for the range $2 \leq R < 4$. Some of the important parameters are listed below. I determined them somewhat arbitrarily after trial and error.

- Difference between R values: 0.0001
- Number of iterations: 2050
- Transient cutoff iteration: 2000

In this diagram, one can approximate where the bifurcations occur. For example, it is obvious that there is a bifurcation from a fixed point to a two-cycle at $R = 3$. Then, at $R \approx 3.45$, the two-cycle doubles to a four-cycle. Further splits are more difficult to discern, but are still apparent. Just before $R = 3.6$, the map devolves into chaos, though there are patches of simpler behavior at greater R values, such as the three-cycle at $R \approx 3.83$, which then sees its own period-doubling cascade before devolving once again.

2 Experimentally Determining the Feigenbaum Number

The Feigenbaum number δ is the ratio by which period-doubling bifurcations occur, *i.e.*

$$\delta = \lim_{n \to \infty} \frac{R_{n+1} - R_n}{R_{n+2} - R_{n+1}} \tag{1}$$

where R_n is the R value at which the n^{th} bifurcation occurs. With this formula, δ can be approximated by finding the first k bifurcation points, $\{R_1, \ldots, R_k\}$, where k is set by the user to achieve a desired accuracy. In Figure 2a, a zoomed-in visualization of the first bifurcation in 1, we can see that $R_1 \approx 2.995$—at this point, the line of dots splits into two diverging paths. As I added more iterations to my model, this number gradually approached $R_1 = 3$, and considering what we know analytically about the logistic map, I let $R_1 = 3$. Then, in 2b, a zoomed-in visualization of the

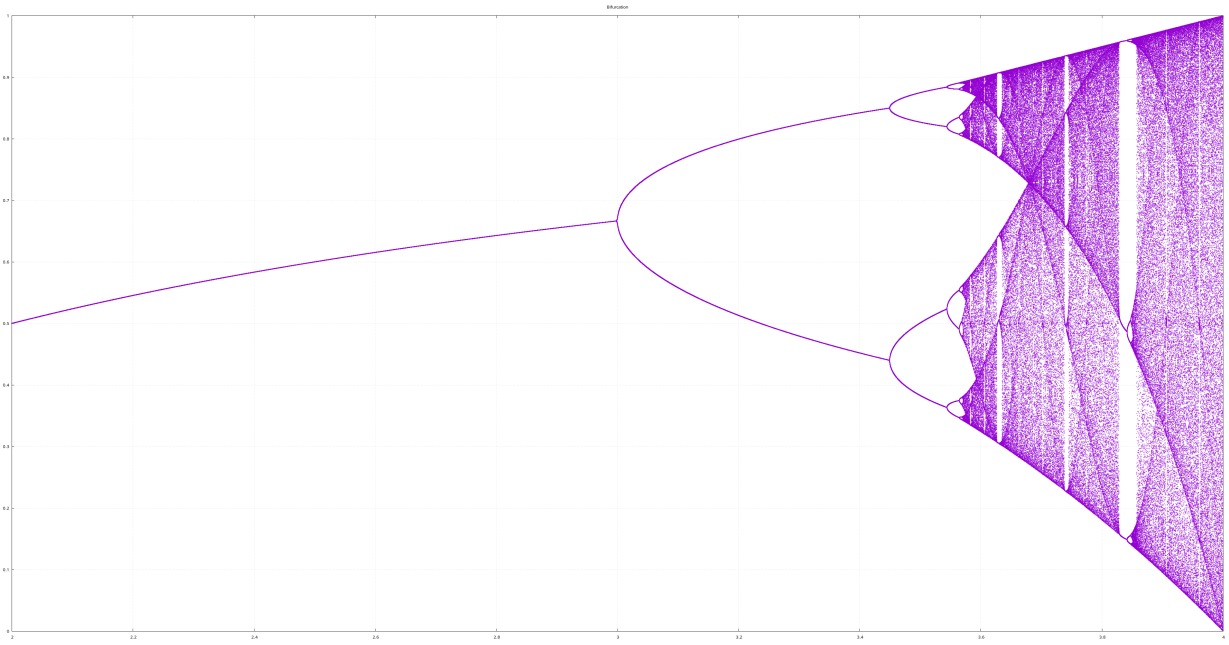


Figure 1: Logistic Map Bifurcation Diagram

second bifurcation of the logistic map, we can see that $R_2 \approx 3.449$.

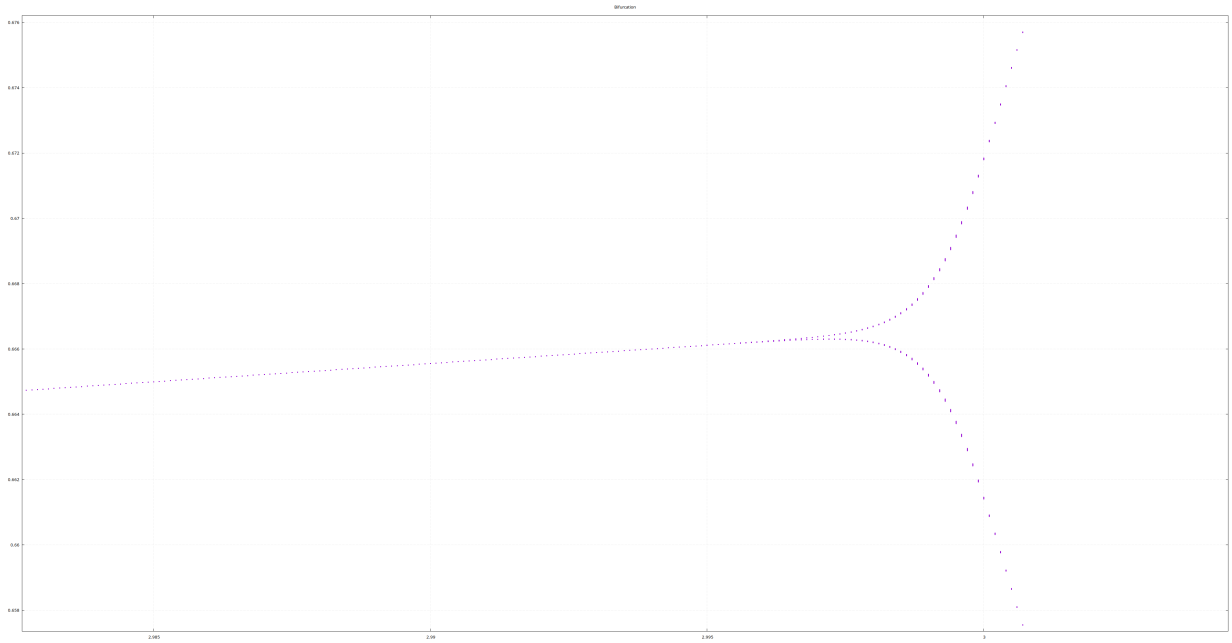
Figure 3a shows that the third bifurcation occurs at $R_3 \approx 3.544$, and 3b shows that the fourth and fifth bifurcations occur at $R_4 \approx 3.565$ and $R_5 \approx 3.5688$, respectively.

Plugging these numbers into Equation 1 and ignoring the limit:

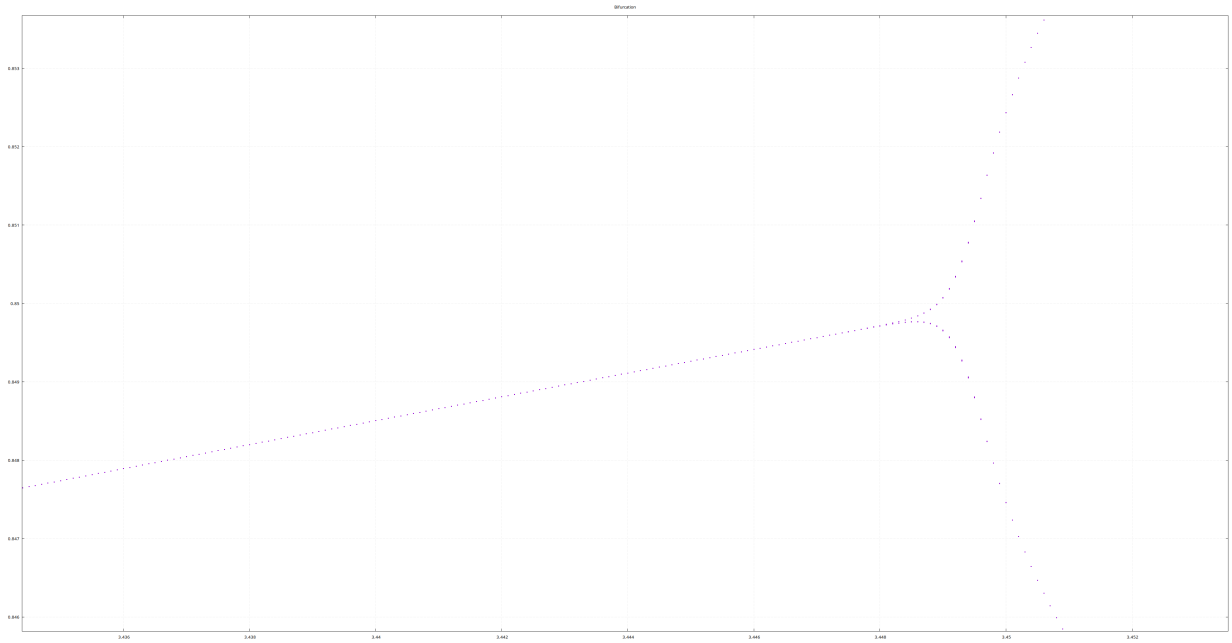
$$\begin{aligned}\delta_1 &= \frac{R_2 - R_1}{R_3 - R_2} \\ &= \frac{3.449 - 3.0}{3.544 - 3.449} \\ &= \frac{0.449}{0.095} \\ &= 4.726315789\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{R_3 - R_2}{R_4 - R_3} \\ &= 4.523809524\end{aligned} \qquad \begin{aligned}\delta_3 &= \frac{R_4 - R_3}{R_5 - R_4} \\ &= 5.526315789\end{aligned}$$

The actual Feigenbaum number is $\delta = 4.669\ldots$, meaning that my δ_1 was the closest approximation. I suspect that the increases in error come from the critical slowing down effect near bifurcation points. Because of the critical slowing down, the bases of the "forks" in the graph occurred slightly before the bifurcation points, obscuring the exact locations. By discarding more transient points and simulating into the tens or hundreds of thousands of iterations, the accuracy would improve, and I could get a more distinct idea of where the bifurcation points actually occur.

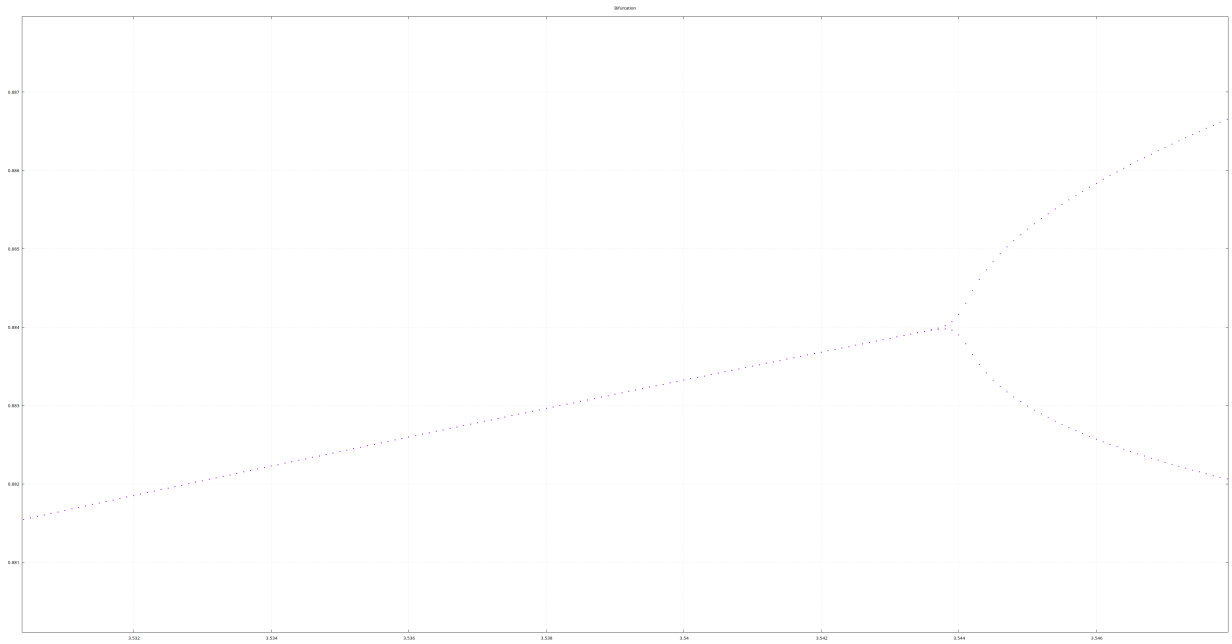


(a) First Bifurcation



(b) Second Bifurcation

Figure 2: First Two Bifurcations of the Logistic Map



(a) Third Bifurcation

(b) Fourth and Fifth Bifurcations

Figure 3: Third, Fourth, and Fifth Bifurcations of the Logistic Map

3 Henon Map Bifurcation Diagram

The Henon map, defined by

$$x_{k+1} = y_k + 1 - ax_k^2$$
$$y_{k+1} = bx_k$$

has a similar bifurcation diagram to the Logistic Map, as shown in Figure 4. This diagram plots the parameter values (a) on the horizontal axis, and the x values on the vertical axis. b is set to 0.3 and the y values are ignored. As in the logistic map, there is an obvious period-doubling cascade, as well as a "break" in the chaos where an odd-numbered cycle appears, this time as a seven-cycle. Once again, the period-doubling cascade can be used to experimentally determine the Feigenbaum number. I omit the zoomed-in images to preserve space, but through my own analysis, I determined the bifurcation points $\{A_1, A_2, ..., A_5\}$ to be

$$A_1 = 0.367$$
$$A_2 = 0.9125$$
$$A_3 = 1.026$$
$$A_4 = 1.0512$$
$$A_5 = 1.0566$$

As before, Equation 1 can be used to approximate the Feigenbaum number:

$$\begin{aligned}\delta_1 &= \frac{A_2 - A_1}{A_3 - A_2} \\ &= \frac{0.9125 - 0.367}{1.026 - 0.9125} \\ &= \frac{0.5455}{0.1135} \\ &= 4.806167401\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{A_3 - A_2}{A_4 - A_3} \\ &= 4.503968254\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{A_4 - A_3}{A_5 - A_4} \\ &= 4.666666667\end{aligned}$$

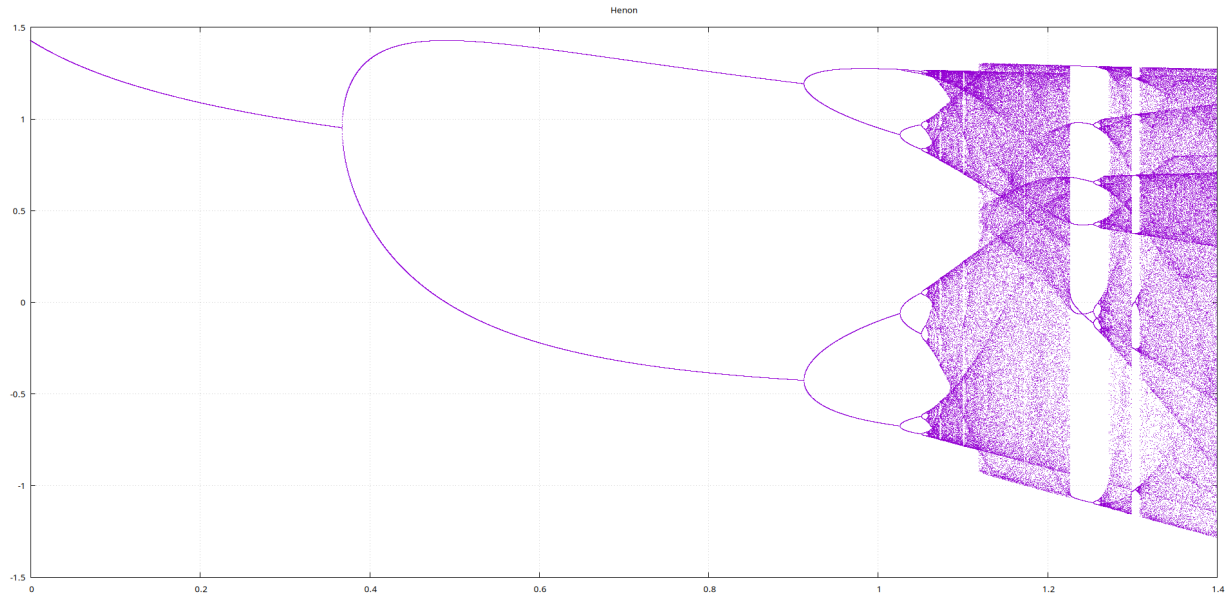


Figure 4: Bifurcation Diagram of the Henon Map

4 Results Analysis

This time, the results were much closer to the actual Feigenbaum number. I used 10 000 iterations instead of 2 000, which clearly improved accuracy, confirming my hypothesis about the critical slowing down effect. (In the future, I will use more iterations, but I want to keep my results from the logistic map experiment to document my process.)

The Feigenbaum number generally applies to one-dimensional maps, but it appears to apply to the Henon map as well, despite it being two-dimensional. Parker and Chua, on pages 206 and 207, describe that Feigenbaum's constant applies to "sufficiently smooth" maps with a quadratic maximum. The Henon map seems to satisfy this condition, as it is very similar to the logistic map, so I omit a formal proof. As such, I would expect that with enough iterations, the experimentally-determined Feigenbaum number would appear out of both the logistic map and the Henon map, *i.e.* my results from parts 2 and 3 should have been the same.

5 Appendix : Code

5.1 logistic_map.cpp

See line 100 for the implementation of the Henon Map.

```cpp
#include "logistic_map.hpp"
#include "plotter.hpp"

int main(int argc, char* argv[]) {

    if (argc != 7) {
        std::cout << "Pass in the right number of parameters" << std::endl;
        return 0;
    }

    double R_min   = std::stod(argv[1]);
    double R_max   = std::stod(argv[2]);
    double R_diff  = std::stod(argv[3]);
    double x0      = std::stod(argv[4]);
    int start_iter = std::stoi(argv[5]);
    int max_iter   = std::stoi(argv[6]);

    int total_iters = max_iter - start_iter;
    VI tvals(total_iters);
    for (int i = 0; i < total_iters; ++i) tvals[i] = i;

    int Rvals_size = static_cast<int>((R_max - R_min) / R_diff) + 1;
    VD Rvals(Rvals_size);
    for (int i = 0; i < Rvals_size; ++i) {
        Rvals[i] = R_min + (i * R_diff);
    }
    cout << "Beginning logistic map simulation \n\n";

    cout << "R min       = " << R_min << "\n"
         << "R_max       = " << R_max << "\n"
         << "R_diff      = " << R_diff << "\n"
         << "x0          = " << x0 << "\n"
         << "max_iter    = " << max_iter << "\n"
         << "start_iter  = " << start_iter << "\n\n"
         << "total_iters = " << total_iters << "\n"
         << "Rvals_size  = " << Rvals_size << "\n";


    VDD bifurcation_data(Rvals_size, VD(total_iters));
    VD yvals(total_iters);

    getHenonBifurcationData(bifurcation_data, yvals, Rvals, x0, 0.5, 0.3, start_iter,
    max_iter);

//    getLogmapBifurcationData(bifurcation_data, Rvals, x0, start_iter, max_iter);

    Plotter::bifurcationPlot(Rvals, bifurcation_data, "Henon");




/*
    logisticMap(xvals, R, x0, max_iter);

    VD plusOneT = extendVec(xvals, R);

    VD plusTwoT = extendVec(plusOneT, R);
```

```
57
58      Plotter::discretePlot(tvals, xvals, "Time-Domain");
59      Plotter::contPlot(xvals, plusOneT, "First-Return-Map");
60      Plotter::contPlot(xvals, plusTwoT, "Second-Return-Map");
61  */
62
63      return 0;
64  }
65
66  void getLogmapBifurcationData(VDD &bifurcation_data, VD &Rvals, double x0, int start_iter,
        int max_iter) {
67      int rvals_size = Rvals.size();
68
69      for (int i = 0; i < rvals_size; ++i) {
70          double R = Rvals[i];
71          logisticMap(bifurcation_data[i], R, x0, start_iter, max_iter);
72      }
73  }
74
75  void getHenonBifurcationData(VDD &bifurcation_data, VD &yvals, VD &Avals, double x0, double
        y0, double b, int start_iter, int max_iter) {
76      int avals_size = Avals.size();
77      for (int i = 0; i < avals_size; ++i) {
78          double a = Avals[i];
79          HenonMap(bifurcation_data[i], yvals, a, b, x0, y0, start_iter, max_iter);
80      }
81  }
82
83  /* Logistic Map */
84
85  void logisticMap(VD &xvals, double R, double x0, int start_iter, int max_iter) {
86
87      double x = x0;
88      int total_iters = max_iter - start_iter;
89
90      for (int i = 0; i < start_iter; ++i) x = R * x * (1 - x);
91
92      for (int i = 0; i < total_iters; ++i) {
93          xvals[i] = x;
94          x = R * x * (1 - x);
95      }
96  }
97
98  /* Henon Map */
99
100 void HenonMap(VD &xvals, VD &yvals, double a, double b, double x0, double y0, int start_iter
        , int max_iter) {
101
102     double x = x0;
103     double y = y0;
104     double x_old = x;
105
106     int total_iters = max_iter - start_iter;
107
108     for (int i = 0; i < start_iter; ++i) {
109         x = y + 1 - (a * x * x);
110         y = b * x_old;
111         x_old = x;
112     }
113
114     for (int i = 0; i < total_iters; ++i) {
115         xvals[i] = x;
116         yvals[i] = y;
117
```

```
        x = y + 1 - (a * x * x);
        y = b * x_old;
        x_old = x;
    }
}
```