Problem Set #1 —— Logistic Map Introduction

University of Colorado - Boulder
CSCI4446 — Chaotic Dynamics

JACK MATHEWS
john.mathewsiii@colorado.edu

January 23, 2026

1 Problem 1

No extra considerations regarding the syllabus.

2 Problem 2

MOOC Materials Completed

3 Problem 3

See Appendix for code listing. Line 51 begins the actual logistic map implementation.

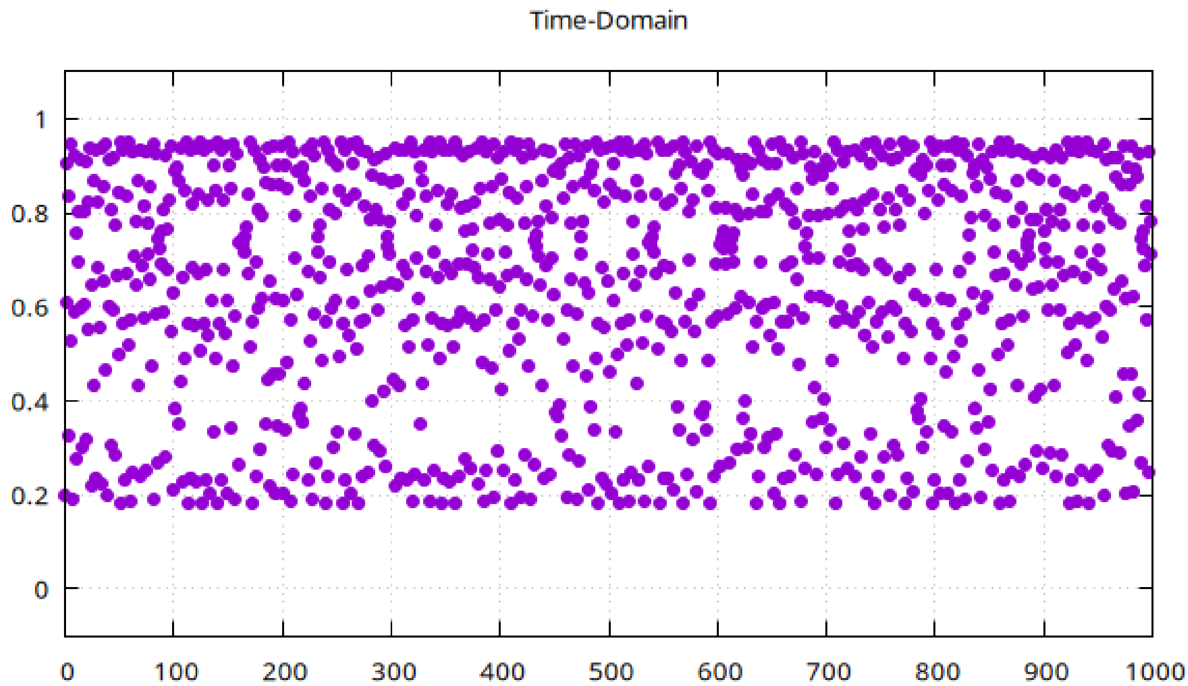
4 Problem 4

4.1 Fixed x_0, variable R

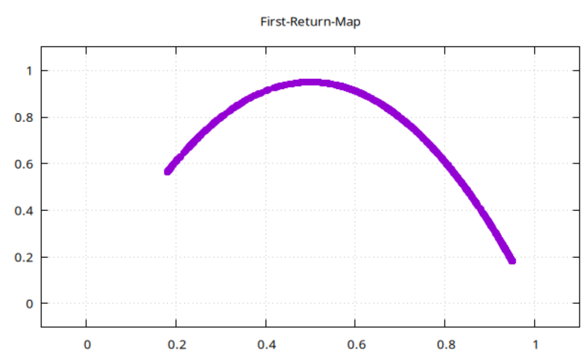
Setting $x_0 = 0.2$ and changing R, we get varied results. For example, in Figure 1, where R is set to 3.8, the results are chaotic. However, for smaller R, such as $R = 2.8$ (Figure 2), the map converges to a fixed point. In between those two, such as when $R = 3.55$, the map converges to a four-cycle after the transient disappears (Figure 3). At every fourth iteration, the map loops, *i.e.* $x_{n+4} \approx x_n$.

With these results, we can conclude that the logistic map has a bifurcation from a fixed point to a limit cycle in the range $(2.8, 3.55)$, though multiple exist in a period-doubling cascade, and a bifurcation from the limit cycle to chaos between $(3.55, 3.8)$.

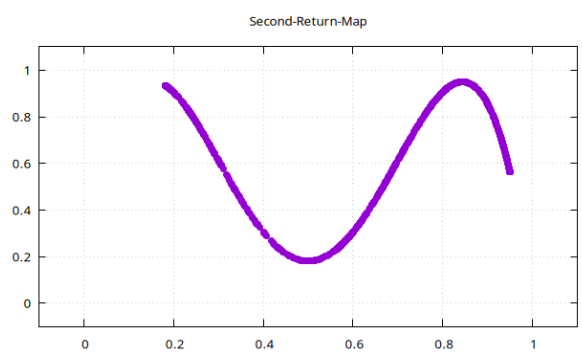
It is worth noting that each of these R values are such that $R < 4$. If $R > 4$, the map diverges to $-\infty$, as in these cases it is possible for $x_{n+1} > 1$ when $x_n < 1$. If $x_{n+1} > 1$, then x_{n+2} will necessarily be negative, and all subsequent iterations will approach negative infinity. If $R = 4$, then when $x_n = 0.5$, $x_{n+1} = 0$, and the system will remain in a null state. All other values of x_n will produce chaotic behavior in x_{n+1}.



(a) Time Domain

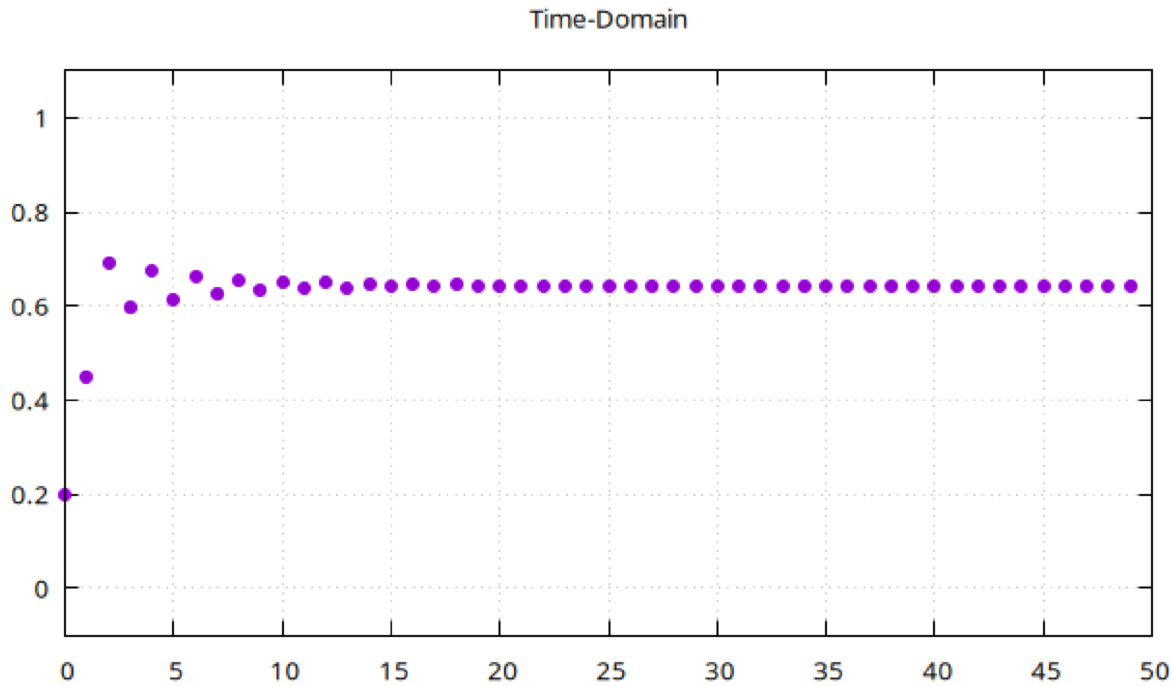


(b) First Return

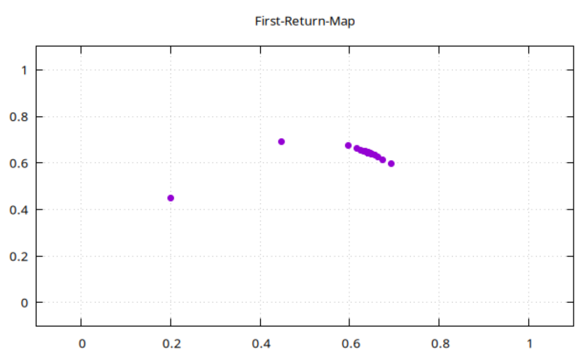


(c) Second Return

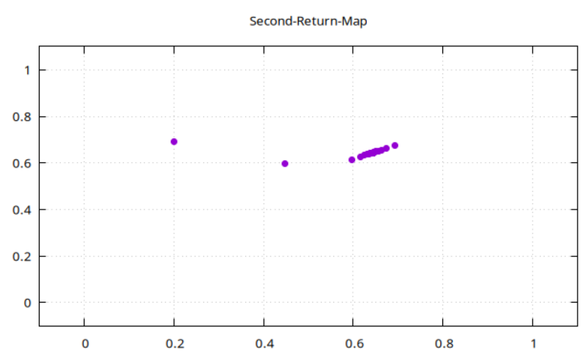
Figure 1: Chaotic Results: $R = 3.8$



(a) Time Domain

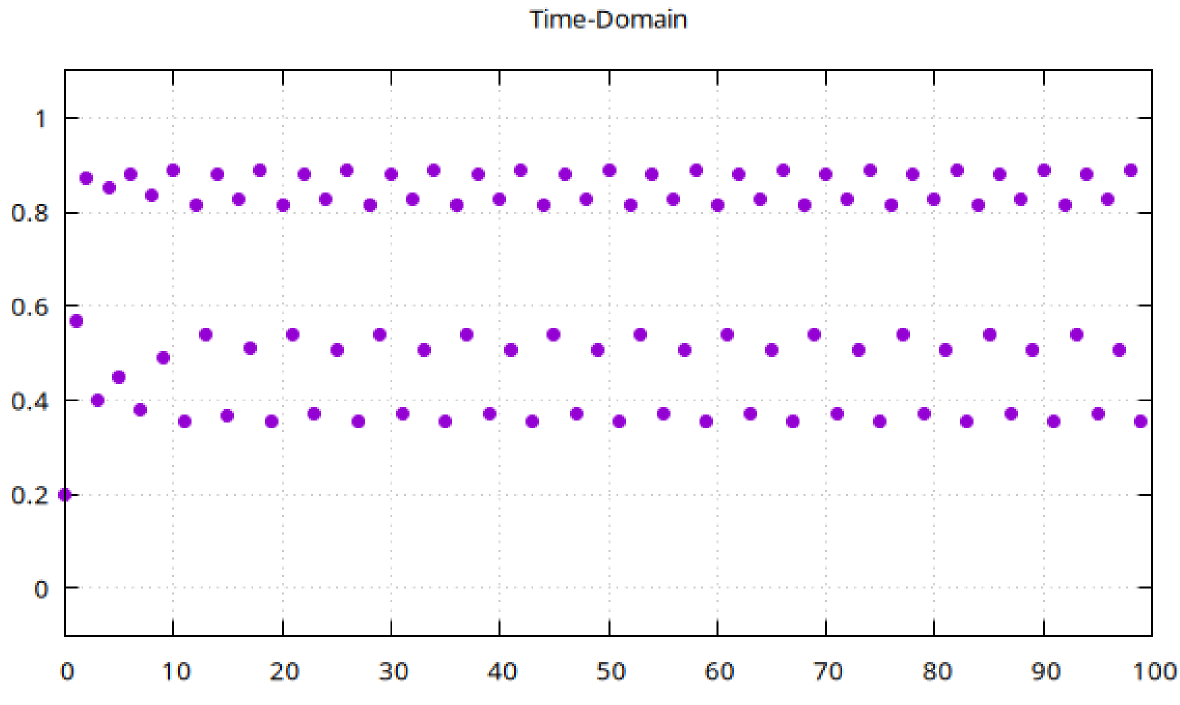


(b) First Return

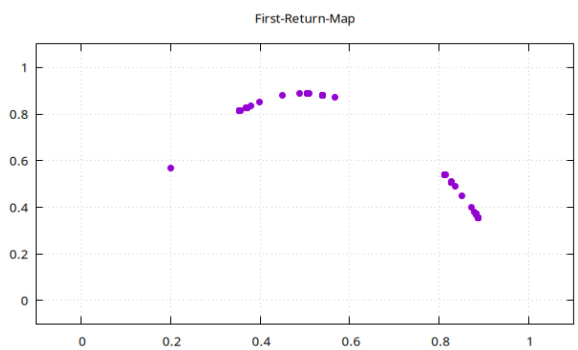


(c) Second Return

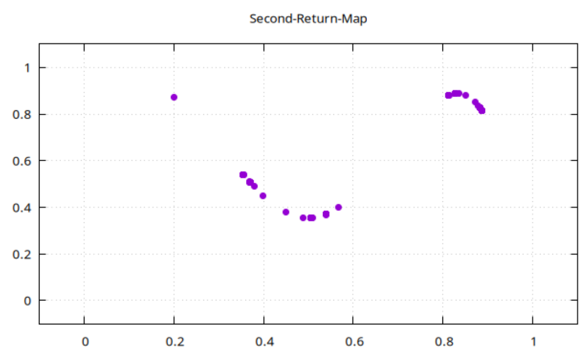
Figure 2: Fixed Point Results: $R = 2.8$



(a) Time Domain



(b) First Return

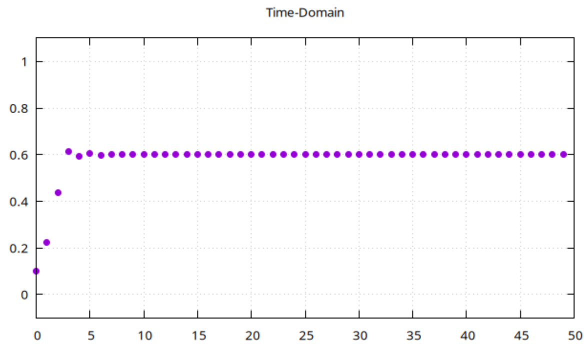


(c) Second Return

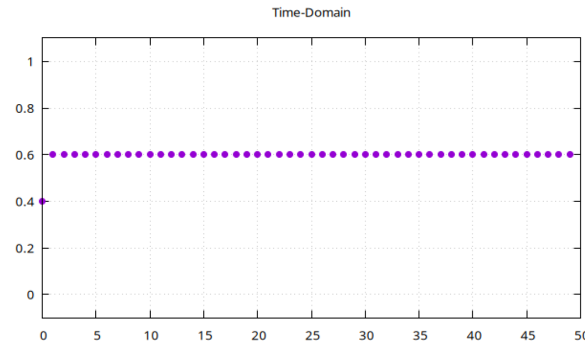
Figure 3: Four-Cycle Results: $R = 3.55$

4.2 Fixed R, variable x_0

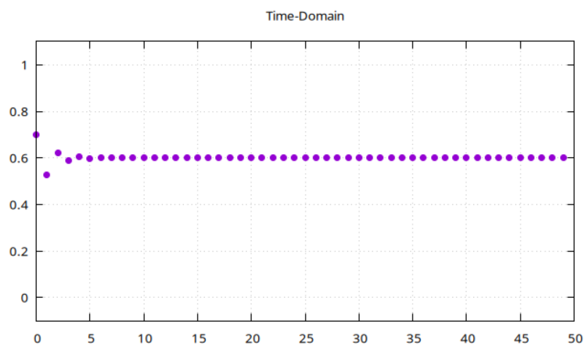
If we fix R to 2.5, changing the initial condition does not affect the overall results. In Figure 4, the logistic map is plotted for $x_0 = 0.1,\ 0.4,\ 0.7,$ and 0.95. In all four cases, the map converges to the same fixed point of $x_p = 0.6$. This set roughly spans $(0, 1)$, suggesting that the map converges to this fixed point regardless of initial condition. This is an example of a basin of attraction that spans the entire domain (global).



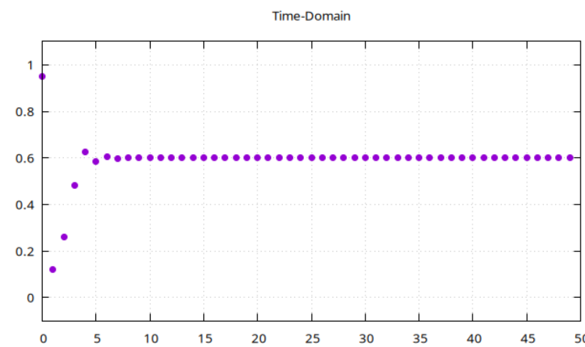
(a) $x_0 = 0.1$



(b) $x_0 = 0.4$



(c) $x_0 = 0.7$



(d) $x_0 = 0.95$

Figure 4: $R = 2.5$ with different initial conditions

5 Appendix: Code

5.1 logistic_map.cpp

```cpp
#include "logistic_map.h"
#include "plotter.hpp"
#include <algorithm>

void logisticMap(VD &xvals, double R, double x0, int max_iter);

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

    if (argc != 4) {
        std::cout << "Pass in the right number of parameters" << std::endl;
        return 0;
    }

    double R = std::stod(argv[1]);
    double x0 = std::stod(argv[2]);
    int max_iter = std::stoi(argv[3]);

    VI tvals;
    tvals.resize(max_iter);

    VD xvals;
    xvals.resize(max_iter);

    for (int i = 0; i < max_iter; ++i) tvals[i] = i;

    logisticMap(xvals, R, x0, max_iter);


    double last = xvals.back();
    double extend = last * R * (1 - last);
    double extend2 = extend * R * (1 - extend);

    VD plusOneT = xvals;
    std::rotate(plusOneT.begin(), plusOneT.begin() + 1, plusOneT.end());
    plusOneT.back() = extend;

    VD plusTwoT = plusOneT;
    std::rotate(plusTwoT.begin(), plusTwoT.begin() + 1, plusTwoT.end());
    plusTwoT.back() = extend2;


    Plotter::discretePlot(tvals, xvals, "Time-Domain");
    Plotter::contPlot(xvals, plusOneT, "First-Return-Map");
    Plotter::contPlot(xvals, plusTwoT, "Second-Return-Map");

    return 0;
}

/* Logistic Map */

void logisticMap(VD &xvals, double R, double x0, int max_iter) {

    double x = x0;
    for (int i = 0; i < max_iter; i++) {
        xvals[i] = x;
        x = R * x * (1 - x);
    }
}
```

```
void printVec(VD &vec) {
    int size = vec.size();
    for (int i = 0; i < size; i++) {
        cout << vec[i] << " ";
        if (i % 5 == 4) {
            cout << "\n";
        }
    }
}
```