Visualizing the logistic map with a microcontroller

Juan D Serna\textsuperscript{1} and Amitabh Joshi\textsuperscript{2}

\textsuperscript{1} School of Mathematical and Natural Sciences, University of Arkansas at Monticello, Monticello, AR 71656, USA
\textsuperscript{2} Department of Physics, Eastern Illinois University, Charleston, IL 61920, USA

E-mail: serna@uamont.edu and ajoshi@eiu.edu

Abstract

The logistic map is one of the simplest nonlinear dynamical systems that clearly exhibits the route to chaos. In this paper, we explore the evolution of the logistic map using an open-source microcontroller connected to an array of light-emitting diodes (LEDs). We divide the one-dimensional domain interval $[0, 1]$ into ten equal parts, an associate and LED to each segment. Every time an iteration takes place a corresponding LED turns on, indicating the value returned by the logistic map. By changing some initial conditions of the system, it is easy to observe the transition from order to chaos exhibited by this map.

Introduction

Nonlinear dynamics is a topic that not only covers all the disciplines, in both natural and social sciences, but is also now becoming part of introductory level undergraduate courses in the sciences. Searching for affordable, easy to set up, and reconfigurable classroom demonstrations that allow us to investigate the physical and mathematical nature of nonlinear dynamical systems has always been a matter of interest for instructors. This paper describes a simple apparatus used to explore the behaviour of one-dimensional chaotic maps, like the logistic map, when different initial conditions are chosen.

The device consists of an inexpensive open-source microcontroller connected to an array of light-emitting diodes (LEDs) and programmed to iterate the logistic map in the one-dimensional interval $[0, 1]$. This interval is divided into ten equal parts and mapped one-to-one to a single LED in the array. When the logistic map produces a certain value after an iteration, the corresponding LED lights up, showing the value in the one-dimensional domain approximately. After several iterations, it is possible to visualize the trajectory of the map by looking at the blinking LEDs sequence. Sensitivity to initial conditions, density of periodic orbits, strange attractors, and bifurcations are visualized easily with this device.

One-dimensional maps

The logistic map is, perhaps, the simplest example of how a nonlinear dynamical equation can give rise to very complex, chaotic behaviour [1]. Initially introduced as a mathematical model of population growth [2], it rapidly found applications in diverse areas like mathematical biology, biometry, demography, condensed matter, econophysics, and computation [3]. The logistic map function is defined as

$$X_{n+1} = AX_n(1 - X_n) \equiv f_A(X),$$

(1)
where the factor $A$ is a model-dependent parameter representing conditions external to the system, and $X_n$ is the population in the $n$th-period cycle [4], scaled so that its value fits in the interval $[0, 1]$. The function $f_A$ is called an iteration function because we may find the population fraction $X$ in the following period cycle by repeating the mathematical operations expressed in (1). It is also one-dimensional since there is only a single variable $X$, and the resulting curve is a line [5].

**Setup**

We show a photograph of the apparatus setup in figure 1, and its schematics in figure 2.

The apparatus uses an open-source Arduino prototyping platform made up of an Atmel AVR processor (microcontroller). It has 14 digital input/output pins, six analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button [6]. Arduino is a registered trademark—only the official boards are named ‘Arduino’—so clones usually have names ending with ‘duino’ [7]. Arduino can be connected to a computer through the USB port and programmed using a language similar to C++. The program is uploaded into the microcontroller using an integrated development environment (IDE). In the Arduino world, programs are known as sketches.

We connected an array of ten 5 mm LEDs to the digital pins of the microcontroller using 220 $\Omega$ resistors (or 1 k$\Omega$ for dimmer LEDs). We also used a solderless breadboard to hook up all the electric components to the microcontroller (see figure 1).

**The sketch**

Box 1 shows a simple program (sketch) used to operate the microcontroller and visualize the logistic map by looking at the array of blinking LEDs. A glowing LED represents an iteration of the one-dimensional map, and it is linked with a value in the interval $[0, 1]$.

Arduino programs require two mandatory functions: `setup()` and `loop()`. Any variable or constant defined outside these two functions is considered to be global. In the `setup()` function, we tell the microcontroller that there are ten LEDs connected to the digital pins and that they are intended to be turned on and off. In the `loop()` function, the logistic map is iterated, and the visualization process takes place as we observe the LEDs turning on and off, one after another, following the evolution of the nonlinear system.

Using the if and else control structures, we divide the interval $[0, 1]$ into ten identical segments and associate an LED with each one of
Box 1. Logistic map sketch for the Arduino.

```c
// blinking logistic map

// choose the pin for each LED
const int NbrLEDs = 10;
const int LEDpin[] = {2,3,4,5,6,7,8,9,10,11};
const int wait = 500; // wait for 500 milliseconds

// logistic map parameters
const double A = 3.7; // Logistic map constant

double X0 = 0.2; // Initial position (0 < X0 < 1.0)
double X = X0; // Use X0 as your first calculated point

// setup() initializes the LED pins
void setup() {
  for (int i = 0; i < NbrLEDs; i++) {
    pinMode(LEDpin[i], OUTPUT);
  }
}

// loop() iterates the logistic map and turn on/off LEDs
void loop() {
  if (X < 0.1) {
    blinkLED(LEDpin[0]);
  } else if ((X >= 0.1) && (X < 0.2)) {
    blinkLED(LEDpin[1]);
  } else if ((X >= 0.2) && (X < 0.3)) {
    blinkLED(LEDpin[2]);
  } else if ((X >= 0.3) && (X < 0.4)) {
    blinkLED(LEDpin[3]);
  } else if ((X >= 0.4) && (X < 0.5)) {
    blinkLED(LEDpin[4]);
  } else if ((X >= 0.5) && (X < 0.6)) {
    blinkLED(LEDpin[5]);
  } else if ((X >= 0.6) && (X < 0.7)) {
    blinkLED(LEDpin[6]);
  } else if ((X >= 0.7) && (X < 0.8)) {
    blinkLED(LEDpin[7]);
  } else if ((X >= 0.8) && (X < 0.9)) {
    blinkLED(LEDpin[8]);
  } else {
    blinkLED(LEDpin[9]);
  }

  // iterates the logistic map function
  X0 = X;
  X = A * X0 * (1.0 - X0);
}

// blinkLED function
// turn on/off LEDs
void blinkLED(const int pin) {
  digitalWrite(pin, HIGH); // turn LED on
  delay(wait); // wait 500 milliseconds
  digitalWrite(pin, LOW); // turn LED off
}
```

For example, the first LED from the left represents the first interval segment [0, 0.1), the second LED represents the segment [0.1, 0.2), and so on. The last LED represents the final segment [0.9, 1.0]. When the microcontroller iterates the logistic map, a value belonging to one of these ten intervals is returned, and the corresponding LED turns on for 500 ms. This process is repeated infinitely, so we can observe the orbit followed by the logistic map by watching the blinking LEDs sequence.

The initial conditions for the logistic map can be changed any time and uploaded again into the microcontroller. Thus, we can explore the behaviour of the chaotic map when different values of the parameters A and X0 are chosen. It is amusing to study the bifurcations of the logistic map by looking at the blinking sequence of LEDs. For example, we can start with the
parameter values $A = 0.9$ and $X_0 = 0.5$ ($0 < A < 1$). The LED associated with the interval $[0.5, 0.6)$ turns on first. As the iterations of the logistic map take place, we observe that the sequence of blinking LEDs moves towards the very first LED, which is associated with the segment $[0, 0.1)$. The sequence ends up with this first LED left on indefinitely. This evolution is shown in figure 3(a), the cobweb diagram of the logistic map for that selection of parameters. For this case, $X = 0$ is an attractor, and the domain $[0, 1]$ forms a basin of attraction.

Another example of an attractor happens when $A = 2.9$ and $X_0 = 0.2$ ($1 < A < 3$). We see that, after a few iterations and blinking LEDs, there is only one LED that remains on indefinitely. This LED corresponds to the interval $[0.6, 0.7)$. In this case, we have a fixed point occurring at $X = 0.655$. As is shown in figure 3(b), the orbit of the map follows a square spiral that converges into that fixed point. Now, if $A = 3.1$ and $X_0 = 0.2$ ($3 < A < 3.44948$), we can observe a two-period cycle; the value of $X$ oscillates between 0.558 and 0.765, and the two corresponding LEDs turn on and off intermittently. Here, the fixed point $X = 0.677$ becomes a repeller. Figure 3(c) shows the periodic behaviour of the logistic map for this set of parameters. Unfortunately, higher periodicity (such as four-period, eight-period cycles, or higher) cannot be observed clearly with this device because of the way the LEDs are mapped into the $[0, 1]$ interval. More LEDs are required to improve the ‘resolution’ of the device. However, the microcontroller has a limit of 13 digital pins, and other electric components (like an extra microchip) need to be incorporated into the circuit if we want more LEDs controlled by the device.

Finally, when $A = 3.7$ and $X_0 = 0.05$ ($3.56994 < A < 4$), chaos onset sets in.
‘unpredictable’ sequence of blinking LEDs is observed. After many iterations, there is no single LED that remains on indefinitely, and no single LED that has not been turned on at least once. The orbit of the logistic map covers all the domain of the interval $[0, 1]$. Figure 3(d) shows all points visited by the logistic map orbit after 10 000 iterations.

Final comments

With this apparatus students may understand more easily the behaviour of one-dimensional chaotic maps looking at an actual one-dimensional array of LEDs; every point in the interval $[0, 1]$ produces another point within the same interval after the logistic map is iterated. By controlling the parameter $A$ and the initial value $X_0$, the instructor and students can observe under what conditions periodic and chaotic behaviour may occur, providing a better understanding of periodicity, bifurcations, and the route to chaos of a nonlinear dynamical system. The setup described here is inexpensive, easy, and fun to assemble; it also enhances computer programming and electronics assembly skills.

Acknowledgment

AJ gratefully acknowledges funding support from RCSA.

Received 21 May 2012

doi:10.1088/0031-9120/47/6/736

References

[1] Weisstein E W, Logistic map From MathWorld—A Wolfram Web Resource http://mathworld.wolfram.com/LogisticMap.html

[2] Verhulst P F 1847 Recherches mathématiques sur la loi d’accroissement de la population Mem. Acad. R. Belg. 20 1–32

[3] Pastijn H 2006 Chaotic growth with the logistic model of P-F Verhulst The Logistic Map and the Route to Chaos: From the Beginnings to Modern Applications ed M Ausloos and M Dirickx (New York: Springer)

[4] Marion J B and Thornton S T 1995 Classical Dynamics of Particles and Systems 4th edn (Fort Worth, TX: Saunders) p 178

[5] Hilborn R C 1994 Chaos and Nonlinear Dynamics 2nd edn (New York: Oxford University Press) p 19

[6] Arduino www.arduino.cc

[7] Schmidt M 2011 Arduino: A Quick-Start Guide (Raleigh: Pragmatic Bookshelf)

[8] Sprott J C 2003 Chaos and Time-Series Analysis (New York: Oxford University Press)