Problem 6.14 Controlling chaos

(a) Write a program that allows the user to turn the control on and off. The trajectory can be seen by plotting x_n versus n. The program should incorporate as input the desired unstable periodic trajectory $x(i)$, the period p, the value of r_0, and the parameter ϵ.

(b) Test your program with $r_0 = 0.95$ and the periods $p = 1, 5$, and 13. Use $\epsilon = 0.02$.

(c) Modify your program so that the values of r as well as the values of x_n are shown. How does r change if we vary ϵ? Try $\epsilon = 0.05, 0.01$, and 0.005.

(d) Add a method to compute n_ϵ, the number of iterations necessary for the trajectory x_n to be within ϵ of $x(1)$ when the control is on. Find $\langle n_\epsilon \rangle$, the average value of n_ϵ, by starting with 100 random values of x_0. Compute $\langle n_\epsilon \rangle$ as a function of ϵ for $\delta = 0.05$, 0.005, 0.0005, and 0.00005. What is the functional dependence of $\langle n_\epsilon \rangle$ on ϵ? ■

6.7 ■ HIGHER-DIMENSIONAL MODELS

So far we have discussed the logistic map as a mathematical model that has some remarkable properties and produces some interesting computer graphics. In this section we discuss some two- and three-dimensional systems that also might seem to have little to do with realistic physical systems. However, as we will see in Sections 6.8 and 6.9, similar behavior is found in realistic physical systems under the appropriate conditions.

We begin with a two-dimensional map and consider the sequence of points (x_n, y_n) generated by

$$x_{n+1} = y_n + 1 - a{x_n}^2 \qquad (6.32a)$$

$$y_{n+1} = bx_n. \qquad (6.32b)$$

The map (6.32) was proposed by Hénon who was motivated by the relevance of this dynamical system to the behavior of asteroids and satellites.

Problem 6.15 The Hénon map

(a) Write a program to iterate (6.32) for $a = 1.4$ and $b = 0.3$ and plot 10^4 iterations starting from $x_0 = 0$, $y_0 = 0$. Make sure you compute the new value of y using the old value of x and not the new value of x. Do not plot the initial transient. Look at the trajectory in the region defined by $|x| \leq 1.5$ and $|y| \leq 0.45$. Make a similar plot beginning from the second initial condition, $x_0 = 0.63135448$, $y_0 = 0.18940634$. Compare the shape of the two plots. Is the shape of the two curves independent of the initial conditions?

(b) Increase the scale of your plot so that all points in the region $0.50 \leq x \leq 0.75$ and $0.15 \leq y \leq 0.21$ are shown. Begin from the second initial condition and increase the number of computed points to 10^5. Then make another plot showing all points in the region $0.62 \leq x \leq 0.64$ and $0.185 \leq y \leq 0.191$. If time permits, make an additional enlargement and plot all points within the box defined by $0.6305 \leq x \leq 0.6325$ and $0.1889 \leq y \leq 0.1895$. You will have to increase the number of computed points to order 10^6. What is the structure of the curves within each box? Does the attractor appear to have a similar structure on smaller and smaller length scales? The region of points from which the points cannot escape is the basin of the Hénon attractor. The attractor is the set of points to which all points in the basin are attracted. That is, two trajectories that begin from different conditions will eventually lie on the attractor.

(c) Determine if the system is chaotic, that is, sensitive to initial conditions. Start two points very close to each other and watch their trajectories for a fixed time. Choose different colors for the two trajectories.

*(d) It is straightforward in principle to extend the method for computing the Lyapunov exponent that we used for a one-dimensional map to higher-dimensional maps. The idea is to linearize the difference (or differential) equations and replace dx_n by the corresponding vector quantity $d\mathbf{r}_n$. This generalization yields the Lyapunov exponent corresponding to the divergence along the fastest growing direction. If a system has f degrees of freedom, it has a set of f Lyapunov exponents. A method for computing all f exponents is discussed in Project 6.24. ■

One of the earliest indications of chaotic behavior was in an atmospheric model developed by Lorenz. His goal was to describe the motion of a fluid layer that is heated from below. The result is convective rolls, where the warm fluid at the bottom rises, cools off at the top, and then falls down later. Lorenz simplified the description by restricting the motion to two spatial dimensions. This situation has been realized experimentally and is known as a Rayleigh–Benard cell. The equations that Lorenz obtained are

$$\frac{dx}{dt} = -\sigma x + \sigma y \qquad (6.33a)$$

$$\frac{dy}{dt} = -xz + rx - y \qquad (6.33b)$$

$$\frac{dz}{dt} = xy - bz, \qquad (6.33c)$$

where x is a measure of the fluid flow velocity circulating around the cell, y is a measure of the temperature difference between the rising and falling fluid regions, and z is a measure of the difference in the temperature profile between the bottom and the top from the normal equilibrium temperature profile. The dimensionless parameters σ, r, and b are determined by various fluid properties, the size of the Raleigh–Benard cell, and the temperature difference in the cell. Note that the variables x, y, and z have nothing to do with the spatial coordinates, but are measures of the state of the system. Although it is not expected that you will understand the relation of the Lorenz equations to convection, we have included these equations here to reinforce the idea that simple sets of equations can exhibit chaotic behavior.

LorenzApp displays the solution to (6.33) using the Open Source Physics 3D drawing framework and is available in the ch06 package. To make three-dimensional plots, we use the Display3DFrame class; the only argument of its constructor is the title for the plot. The following code fragment sets up the plot.

```
Display3DFrame frame = new Display3DFrame("Lorenz attractor");
Lorenz lorenz = new Lorenz();
frame.setPreferredMinMax(-15.0, 15.0, -15.0, 15.0, 0.0, 50.0);
frame.setDecorationType(VisualizationHints.DECORATION_AXES);
frame.addElement(lorenz); // lorenz is a 3D element
```