1 Lorenz Equations

Lab Objective: *Investigate the behavior of a system that exhibits chaotic behavior. Demonstrate methods for visualizing the evolution of a system.*

Chaos is everywhere. It can crop up in unexpected places and in remarkably simple systems, and a great deal of work has been done to describe the behavior of chaotic systems. One primary characteristic of chaos is that small changes in initial conditions result in large changes over time in the solution curves.

The Lorenz System

One of the earlier examples of chaotic behavior was discovered by Edward Lorenz. In 1963, while working to study atmospheric dynamics he derived the simple system of equations

$$\begin{aligned}\frac{\partial x}{\partial t} &= \sigma\,(y - x)\\ \frac{\partial y}{\partial t} &= \rho x - y - xz\\ \frac{\partial z}{\partial t} &= xy - \beta z\end{aligned}$$

where σ, ρ, and β are all constants. After deriving these equations, he plotted the solutions and observed some unexpected behavior. For appropriately chosen values of σ, ρ, and β, the solutions did not tend toward any steady fixed points, nor did the system permit any stable cycles. The solutions did not tend off toward infinity either. With further work, he began the study of what was called a *strange attractor*. This system, though relatively simple, exhibits chaotic behavior.

Problem 1. Write a function that implements the Lorenz equations. Let $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$. Make a 3D plot of a solution to the Lorenz equations for an initial condition where (x, y, z) are drawn randomly from a uniform distribution from -15 to 15. As usual, use `scipy.integrate.odeint` to compute the solution. Compare your results with Figure 1.1

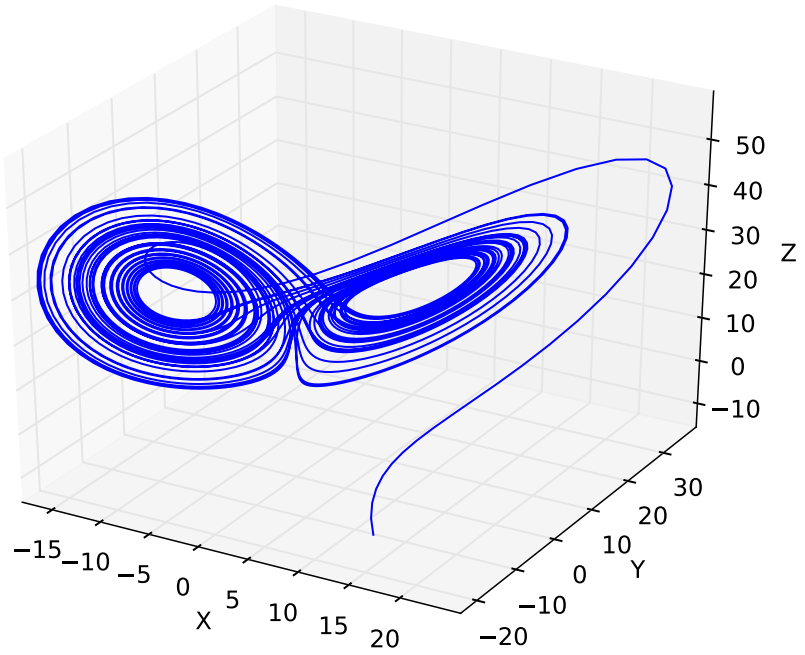

Figure 1.1: Approximate solution to the Lorenz equation with random initial conditions

Basin of Attraction

Notice in the first problem that the solution tended to a 'nice' region. This region is a basin of attraction, and the set of numerical values towards which a system will converge to is an **attractor**. Consider what happens when we change up the initial conditions.

Problem 2. To better visualize the Lorenz attractor, produce a single 3D plot displaying three solutions to the Lorenz equations, each with random initial conditions. Compare your results with Figure 1.2

Chaos

Chaos systems exhibit high sensitivity to initial conditions. This means that a small difference in initial conditions may still result in solutions that are largely different over time. Chaotic systems are not *random*. According to Lorenz, chaos is "[w]hen the present determines the future, but the approximate present does not approximately determine the future."

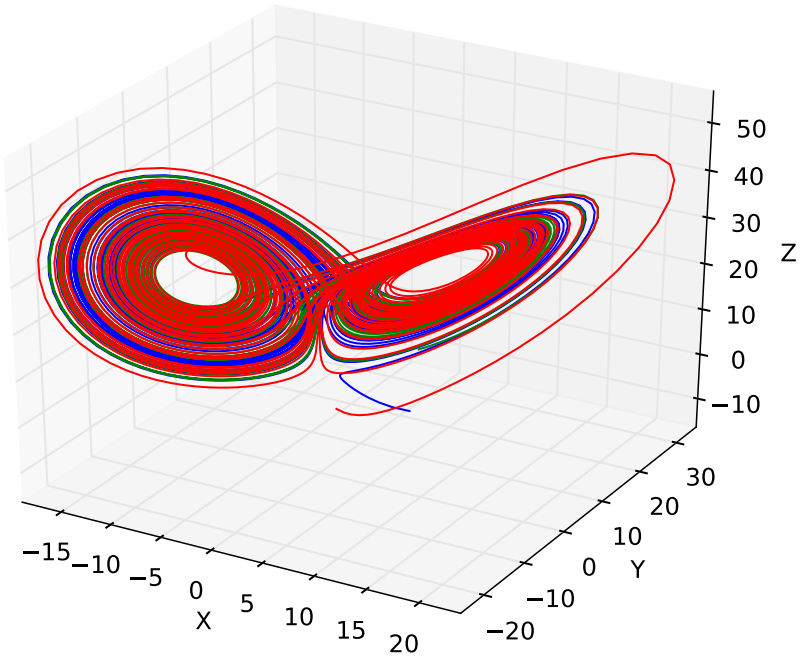

Figure 1.2: Multiple solutions to the Lorenz equation with random initial conditions

Problem 3. Use `matplotlib.animation.FuncAnimation` to produce a 3D animation of two solutions to the Lorenz equations with similar initial conditions. To make similar initial conditions, draw (x_1, y_1, z_1) randomly as before, and then produce (x_2, y_2, z_2) by adding a small perturbation: `np.random.randn(3)*(1e-10)`. It will take several seconds before the separation between the two solutions will be noticeable.

The animation should have a point marker and the past trajectory curve for each solution. Save your animation as `lorenz_animation1.mp4`.

In a chaotic system, round-off error implicit in a numerical method can also produce similarly diverging approximations *for the same initial condition*. For example, using a Runge-Kutta method with two different values of h on identical initial conditions will still result in approximations that differ in a chaotic fashion.

Problem 4. The `odeint` function allows users to specify error tolerances (similar to setting a value of h for a Runge-Kutta method). Using a single random initial condition, produce two approximations by using the `odeint` arguments (`atol=1e-14`, `rtol=1e-12`) for the first

approximation and (`atol=1e-15`, `rtol=1e-13`) for the second.

As in the previous problem, use `FuncAnimation` to animate both solutions. Save the animation as `lorenz_animation2.mp4`.

Lyapunov Exponents

The *Lyapunov exponent* of a dynamical system is one measure of how chaotic a system is. While there are more conditions for a system to be considered chaotic, one of the primary indicators of a chaotic system is *extreme sensitivity to initial conditions*. Strictly speaking, this is saying that a chaotic system is poorly conditioned. In a chaotic dynamical system, the sensitivity to changes in initial conditions depends exponentially on the time the system is allowed to evolve. If $\delta(t)$ represents the difference between two solution curves, when $\delta(t)$ is small, the following approximation holds.

$$\|\delta(t)\| \sim \|\delta(0)\|e^{\lambda t}$$

where λ is a constant called the Lyapunov exponent. In other words, $log(\|\delta(t)\|)$ is approximately linear as a function of time, with slope λ. For the Lorenz system (and the parameter values specified in this lab), it can be verified experimentally that $\lambda \approx .9$.

Problem 5. Estimate the Lyapunov exponent of the Lorenz equations by doing the following:

- Produce an initial condition that already lies on the attractor. This can be done by picking a random initial condition, and then solving the system forward in time for a while.
- Produce a second initial condition by adding a small perturbation: `np.random.randn(3)*(1e-10)`.
- For both initial conditions, use `odeint` to produce approximate solutions for $0 \leq t \leq 10$.
- Compute $\|\delta(t)\|$ by taking the norm of the difference between the two approximate solutions.
- Use `scipy.stats.linregress` to calculate a best-fit line for $log(\|\delta(t)\|)$ against t.
- The slope of the resulting best-fit line is an approximation of the Lyapunov exponent λ.

Print your estimate for λ. Also use plt.semilogy to produce a semilog plot of $\|\delta(t)\|$ and the best-fit line you calculated. Compare your results to Figure 1.3.

Hint: Remember that the best-fit line you calculated corresponds to a best-fit exponential for $\|\delta(t)\|$. If `a` and `b` are the slope and intercept of the best-fit line, respectively, the best-fit exponential can be plotted on a semilog plot using `plt.semilogy(t,np.exp(a*t+b))`.

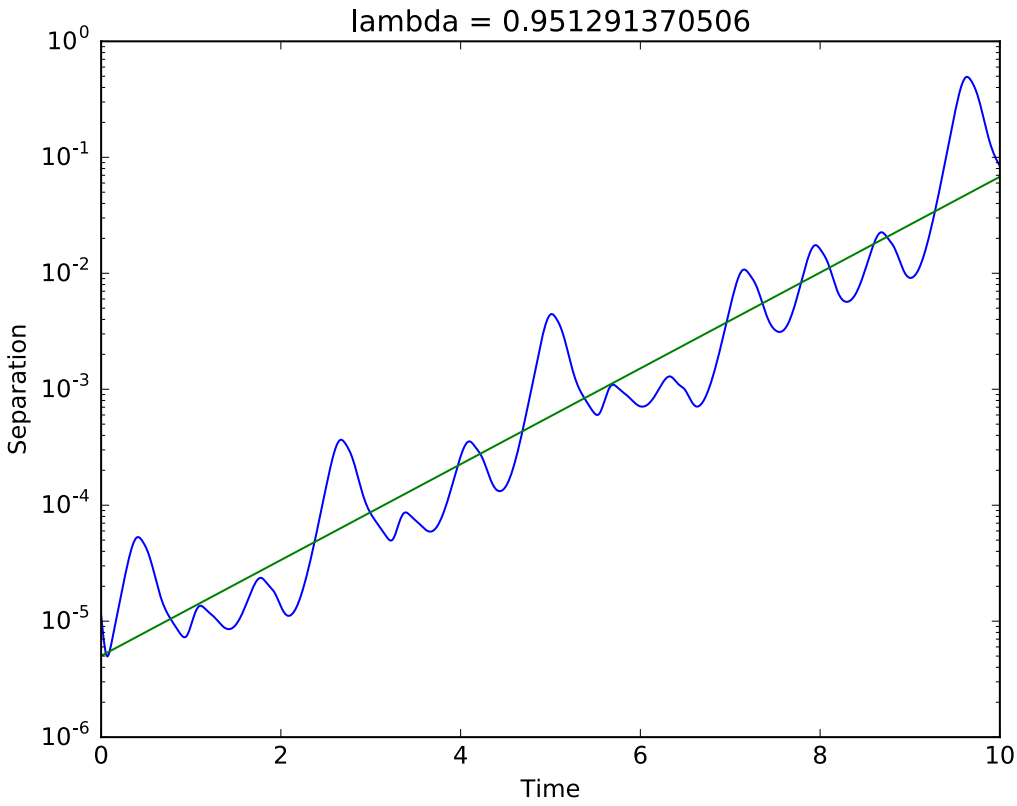

Figure 1.3: A semilog plot of the separation between two solutions to the Lorenz equations together with a fitted line that gives a rough estimate of the Lyapunov exponent of the system.