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

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

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, where the initial conditions x_0, y_0, z_0 are each drawn randomly from a uniform distribution on $[-15, 15]$. As usual, use `scipy.integrate.odeint` to compute an approximate solution. Compare your results with Figure 1.1.

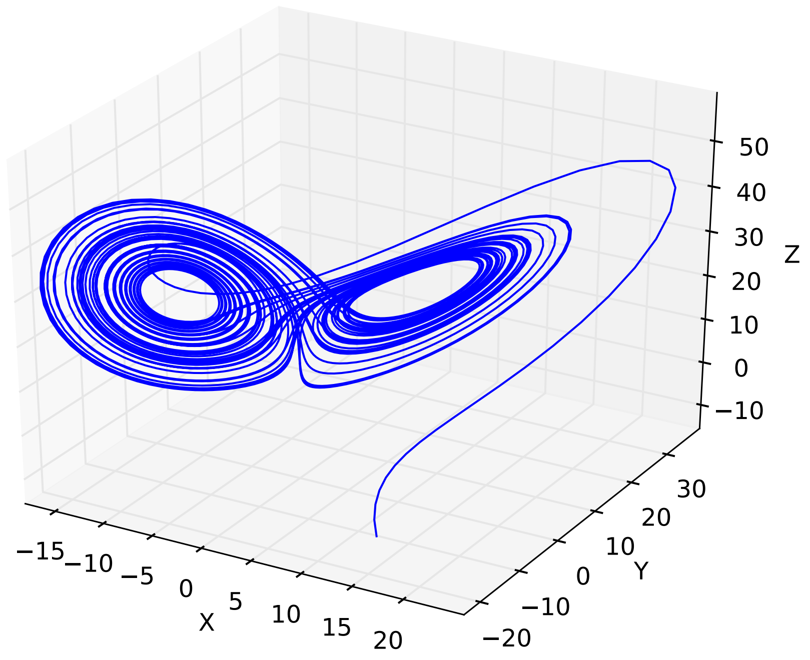

Figure 1.1: Approximate solution to the Lorenz equation with random initial conditions

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 (as before). Compare your results with Figure 1.2.

Chaos

Chaotic systems exhibit high sensitivity to initial conditions. This means that a small difference in initial conditions may result in solutions that diverge significantly from each other. However, chaotic systems are not *random*. According to Lorenz, chaos is "when the present determines the future, but the approximate present does not approximately determine the future."

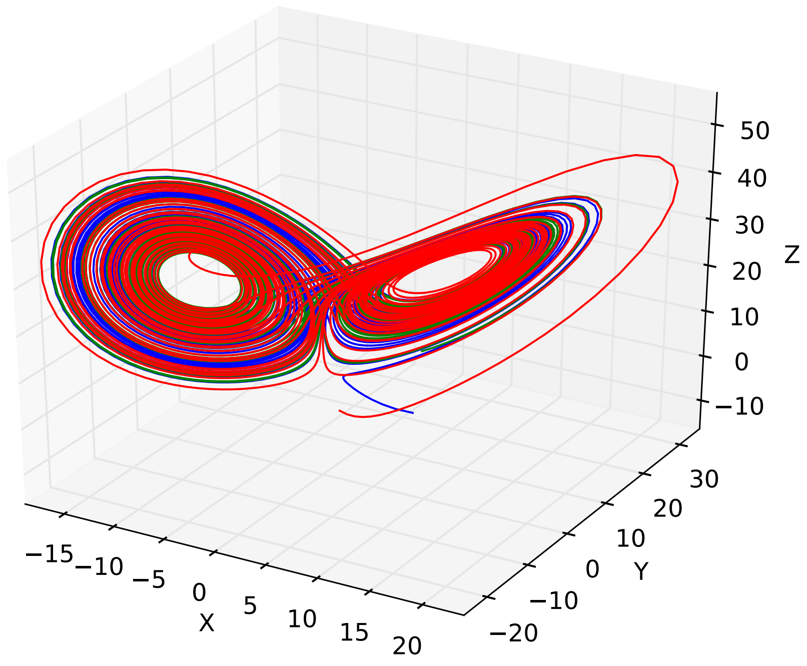

Figure 1.2: Multiple solutions to the Lorenz equation with random initial conditions

Problem 3. Use `matplotlib.animation.FuncAnimation` to produce a 3D animation of two solutions to the Lorenz equations with nearly identical initial conditions. To make the initial conditions, draw x_0, y_0, z_0 as before, and then make a second initial condition by adding a small perturbation to the first (Hint: try using `np.random.randn(3)*(1e-10)` for the perturbation). Note that it may take several seconds before the separation between the two solutions will be noticeable.

The animation should display a point marker as well as the past trajectory curve for each solution. Save your animation as `lorenz_animation.mp4`.

In a chaotic system, round-off error implicit in a numerical method can also cause divergent solutions. For example, using a Runge-Kutta method with two different values for the stepsize h on identical initial conditions will still result in approximations that differ in a chaotic fashion.

Problem 4. The `odeint` function allows user to specify error tolerances (similar to setting a value of h in a Runge-Kutta method). Using a single initial condition, produce two approximations by using the `odeint` arguments (`atol=1e-15, rtol=1e-13`) for the first approximation and (`atol=1e-12, rtol=1e-10`) for the second.

As in the previous problem, use `FuncAnimation` to animation 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 system, the sensitivity to changes in initial conditions depends expoentially 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 for the parameter values specified in this lab), experimentally it can be verified 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 in the attractor. This can be done by using a random "dummy" initial condition, approximating the resulting solution to the Lorenz system for a short time, and then using the endpoint of that solution (which is now in the attractor) as the desired initial condition.
- Produce a second initial condition by adding a small perturbation to the first (as before).
- 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 vector difference between the two solutions for each value of t.
- Use `scipy.stats.linregress` to calculate a best-fit line for $\log(\|\delta(t)\|)$ against t.
- The slope of the best-fit line is an approximation for the Lyapunov exponent λ

Produce a plot similar to Figure 1.3 by using `plt.semilogy`.

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, the best-fit exponential can be plotted using `plt.semilogy(t,np.exp(a*t+b)`.

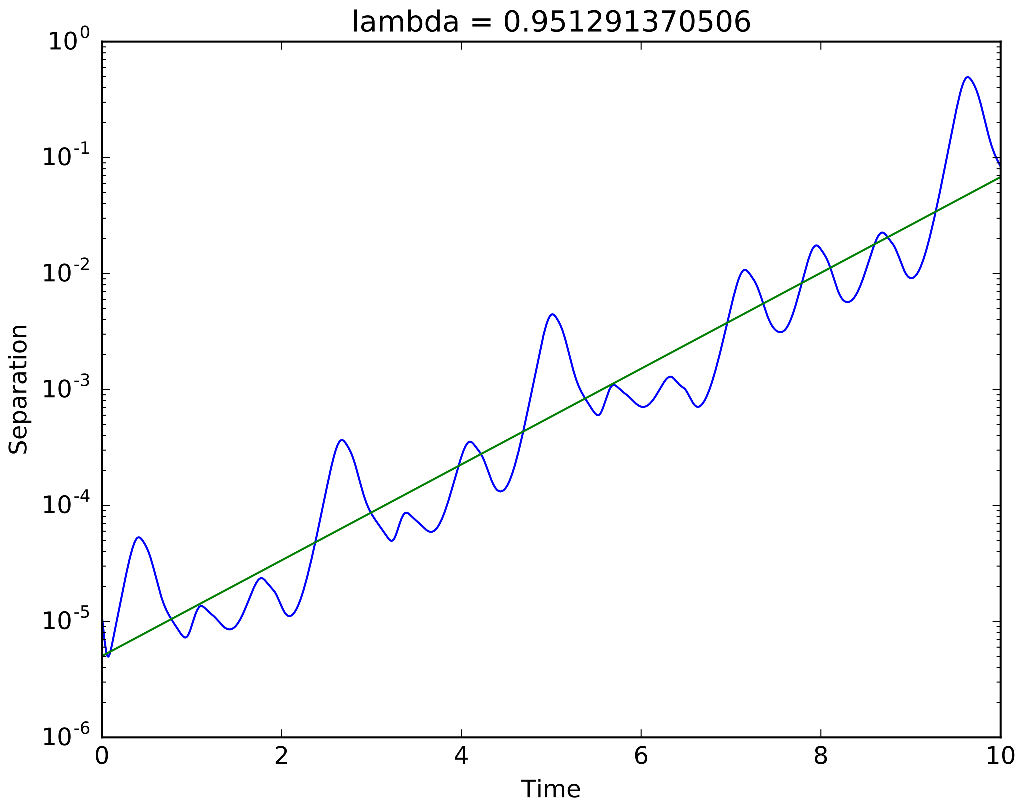

Figure 1.3: A semilog plot of the separation between two solutions to the Lorenz equations together with a fitted line that gives a rough estimate of the Lyapunov exponent of the system.