Computer Project 9
The Lorenz Attractor and Chaotic Dynamics

DUE: April 06, 2023

Introduction: In 1961, mathematician/meteorologist Edward Lorenz came up with a simplified model for making certain weather predictions. The equations he considered have come to be known as The Lorenz Equations and have the form

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x)\\ \frac{dy}{dt} &= \rho x - y - xz\\ \frac{dz}{dt} &= -\beta z + xy,\end{aligned}$$

where the parameters σ, ρ, and β are positive real numbers. In studying the numerical solutions to these equations, Lorenz made a remarkable discovery. If we pick two sets of initial data that are relatively close together, the solutions launched by these two different initial data will quickly look very different. This phenomenon is known as "sensitive dependence on initial conditions" and is a hallmark of chaotic dynamical systems. In essence, sensitive dependence on initial conditions means that long term prediction of the behavior of a complex system is extremely difficult (since we only know the initial conditions to some finite degree of accuracy).

For this assignment take $\sigma = 10$, $\beta = \frac{8}{3}$, and $\rho = 45$.

1) Find all equilibria for the system with the given values of σ, β, and ρ.

2) For each equilibrium, find the linearization of the system. What are the eigenvalues for each matrix?

3) Use your implementation of `vecRK4` from Project 7 to find a numerical solution to the Lorenz Equations with initial conditions $x(0) = 0, y(0) = 1.85$, and $z(0) = 0$ on the interval $0 \leq t \leq 50$. Give a plot of all three functions $x(t), y(t)$, and $z(t)$ (on separate plots). I used 100000 steps for each run.

4) Repeat 3), but with the initial conditions $x(0) = 0.01, y(0) = 1.86$, and $z(0) = -0.01$.

5) Produce a plot where you compare the function $x(t)$ from 3) to $x(t)$ from 4). Does knowledge of $x(t)$ from 3) help to make long-term predictions about the $x(t)$ from 4)? Do the same for $y(t)$ and $z(t)$.

6) Produce a three–dimensional parametric plot of the solution from 3). Your trajectory should produce a tracing of the famous Lorenz Attractor. Sample code for a different chaotic system appears below.

```
#define parameters
a = 0.35
b = 0.1
c = 5.7

#define system for Rossler Attractor
def Rossler(t, vec):
   ret = np.zeros_like(vec)
   ret[0] = -vec[1] - vec[2]
   ret[1] = vec[0] + a*vec[1]
   ret[2] = b + vec[0]*vec[2] - c*vec[2]
   return ret

# Run RK-4 for the Rossler Attractor
T, Ret = vecRK4(Rossler, [1,1,1], 0, 100, 100000)

# Run RK-4 for the Rossler Attractor (slightly different Init Data)
T, Retb = vecRK4(Rossler, [1.1,0.8,0.9], 0, 100, 100000)

# Plot x, y, z for both solutions
ax1 = plt.subplot(311)
plt.plot(T,Ret[0], color='blue')
plt.plot(T,Retb[0], color='red')
plt.xlabel('t')
plt.ylabel('x')

ax2 = plt.subplot(312)
plt.plot(T,Ret[1], color='blue')
plt.plot(T,Retb[1], color='red')
plt.xlabel('t')
plt.ylabel('y')

ax3 = plt.subplot(313)
plt.plot(T,Ret[2], color='blue')
plt.plot(T,Retb[2], color='red')
plt.xlabel('t')
plt.ylabel('z')
plt.show()

#Plot 3D parametric plot for the Rossler Attractor
ax = plt.figure().add_subplot(projection='3d')
ax.plot(Ret[0], Ret[1], Ret[2], label='Rossler Attractor',color='red')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```

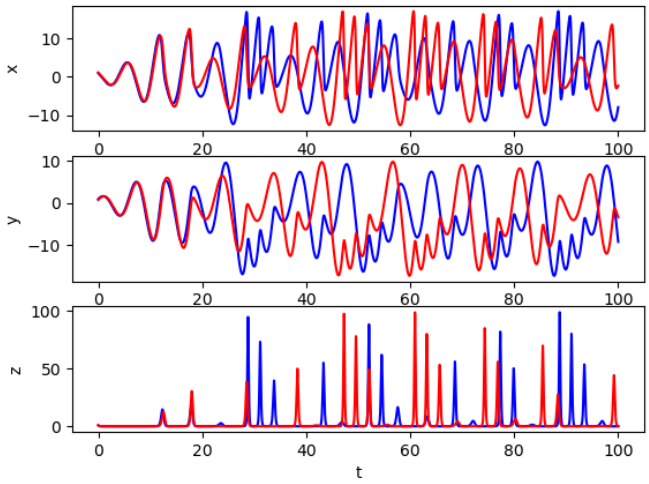

Figure 1: Plots of x, y, and z vs. t for Similar Initial Data

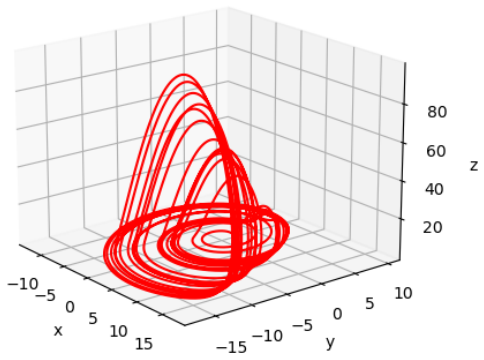

Figure 2: Plot of the Rössler Attractor