Simulation of The Rössler Attractor as a Chaotic system

Amir M. Gholizad

Department of Physics
AmirKabir University of Technology

Abstract

In this project, we have discussed an example of the chaotic behavior of a particular system. The Rössler attractor is a good example of a chaotic system that we have chosen for this project. We will study some basic dynamical properties such as chaotic behavior of the attractor, sensitivity analysis, variation of parameter, bifurcation diagram and Poincar´e map. We presented an explicit Python algorithm for solving the Rössler system and to analyze its dynamical properties.

Keywords: Rössler attractor, Runge-Kutta method, chaotic system, ODE

1. The Rössler attractor

The Rössler attractor is a system of three non-linear ordinary differential equations originally studied by *Otto Rössler*. These differential equations define a continuous-time dynamical system that exhibits chaotic dynamics associated with the fractal properties of the attractor.

The Rössler system is defined by these three ODEs

$$\dot{x} = -y - z \qquad [1]$$

$$\dot{y} = x + ay \qquad [2]$$

$$\dot{z} = b + z(x - c) \qquad [3]$$

where a, b and c are the parameters.

As you can see, there is a non-linear term in the system, zx. If we decide to solve this set of ODEs using analytical tools, we should probably approach it as an Eigenvalue problem and try to calculate the eigenvalues and eigenvectors of the Jacobian matrix. But we can use a known numerical method, The Runge-Kutta method of order 4.

2. The Runge-Kutta method

In numerical analysis, the Runge–Kutta methods are a family of implicit and explicit iterative methods, which include the well-known routine called the Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations. These methods were developed around 1900 by the German mathematicians, Carl Runge and Wilhelm Kutta.

The most widely known member of the Runge–Kutta family is generally referred to as "RK-4", the "classic Runge–Kutta method" or simply as “The Runge–Kutta method".

RK-4 is capable of solving only first-order ODEs, but after applying some modifications, we can turn a second-order ODE into two first order ODEs and solve them with the RK-4 method. It is also useful when we want to solve a set of coupled ODEs. Let’s assume that we have a set of coupled differential equations like

$$\frac{dx}{dt} = f_1(x, y, z, t) \qquad [4]$$

$$\frac{dy}{dt} = f_2(x, y, z, t) \qquad [5]$$

$$\frac{dz}{dt} = f_3(x, y, z, t) \qquad [6]$$

First, we need to have an interval such as $[t_0, t_N]$,which is the interval that we want to use for plotting the answer. Then we will need the initial value of each equation ($x_0 = x(t_0),\ y_0 = y(t_0)\ \ and\ \ z_0 = z(t_0)$). Now that we have our interval, we should discretize it to N separate numbers. The distance between a number and its nearest neighbors will be

$$h = \frac{t_N - t_0}{N} \qquad [7]$$

Instead of continuous t variables, now we have an array of t_ns. Each t_n will be defined as

$$t_n = t_0 + n \cdot h \qquad , \ n = 0,1,2,3, \dots \qquad [8]$$

We calculate every x, y and z functions by using this iterations

$$x_n = x_{n-1} + h\left(\frac{k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x}}{6}\right) \qquad [9]$$

$$y_n = y_{n-1} + h\left(\frac{k_{1y} + 2k_{2y} + 2k_{3y} + k_{4y}}{6}\right) \qquad [10]$$

$$z_n = z_{n-1} + h\left(\frac{k_{1z}+2k_{2z}+2k_{3z}+k_{4z}}{6}\right) \quad [11]$$

where

- k_1 is the slope at the beginning of the interval.
- k_2 and k_3 are the slopes at the midpoints of the interval.
- k_4 is the slope at the end of the interval.

Each iteration, we calculate the k_is for both x, y and z.

This is how we calculate slopes

$$\begin{cases} k_{1x} = f_1(x_{n-1}, \mathrm{y}_{n-1}, z_{n-1}, \mathrm{t}_{n-1}) \\ k_{1y} = f_2(x_{n-1}, \mathrm{y}_{n-1}, z_{n-1}, \mathrm{t}_{n-1}) \\ k_{1z} = f_3(x_{n-1}, \mathrm{y}_{n-1}, z_{n-1}, \mathrm{t}_{n-1}) \end{cases} \quad [12]$$

$$\begin{cases} k_{2x} = f_1\left(x_{n-1} + k_{1x}\cdot\frac{h}{2}, y_{n-1} + k_{1x}\cdot\frac{h}{2}, z_{n-1} + k_{1x}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \\ k_{2y} = f_2\left(x_{n-1} + k_{1y}\cdot\frac{h}{2}, y_{n-1} + k_{1y}\cdot\frac{h}{2}, z_{n-1} + k_{1y}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \\ k_{2z} = f_3\left(x_{n-1} + k_{1z}\cdot\frac{h}{2}, y_{n-1} + k_{1z}\cdot\frac{h}{2}, z_{n-1} + k_{1z}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \end{cases} \quad [13]$$

$$\begin{cases} k_{3x} = f_1\left(x_{n-1} + k_{2x}\cdot\frac{h}{2}, y_{n-1} + k_{2x}\cdot\frac{h}{2}, y_{n-1} + k_{2x}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \\ k_{3y} = f_2\left(x_{n-1} + k_{2y}\cdot\frac{h}{2}, y_{n-1} + k_{2y}\cdot\frac{h}{2}, y_{n-1} + k_{2y}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \\ k_{3z} = f_3\left(x_{n-1} + k_{2z}\cdot\frac{h}{2}, y_{n-1} + k_{2z}\cdot\frac{h}{2}, y_{n-1} + k_{2z}\cdot\frac{h}{2}, \mathrm{t}_{n-1} + \frac{h}{2}\right) \end{cases} \quad [14]$$

$$\begin{cases} k_{4x} = f_1(x_{n-1} + k_{3x}\cdot h, y_{n-1} + k_{3x}\cdot h, y_{n-1} + k_{3x}\cdot h, \mathrm{t}_{n-1} + h) \\ k_{4y} = f_2(x_{n-1} + k_{3y}\cdot h, y_{n-1} + k_{3y}\cdot h, y_{n-1} + k_{3y}\cdot h, \mathrm{t}_{n-1} + h) \\ k_{4z} = f_3(x_{n-1} + k_{3z}\cdot h, y_{n-1} + k_{3z}\cdot h, y_{n-1} + k_{3z}\cdot h, \mathrm{t}_{n-1} + h) \end{cases} \quad [15]$$

3. Analyzing the results

We know the algorithm, now we can apply it to Equations 1, 2 and 3. To solve these equations we should have a, b and c variables and also initial conditions.

The initial ODEs take the form

$$\frac{dx}{dt} = -y - z = f_1(x, y, z, t) \qquad [16]$$

$$\frac{dy}{dt} = x + ay = f_2(x, y, z, t) \qquad [17]$$

$$\frac{dz}{dt} = b + z(x - c) = f_3(x, y, z, t) \qquad [18]$$

We can enter these algorithms into a programming environment, in this project PyCharm, and see the plots for different values of variables.

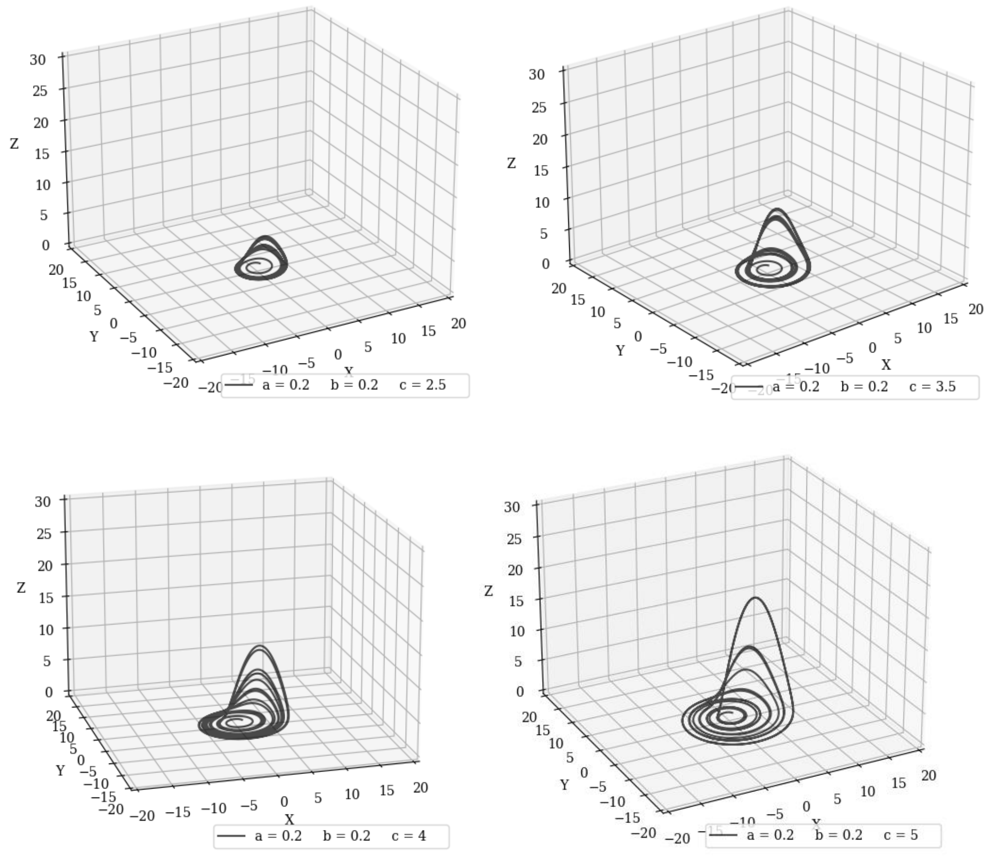

Figure 1

Figure 1 shows three-dimensional projections of the system's attractor for different values of c (with $a = b = 0.2$). As it is shown, by growing c value and keeping

a and b fixed, the number of limit cycles increases, and also it becomes more dependent on z value. These graphs have been plotted based on the initial condition $x_0 = 1, y_0 = 1$ and $z_0 = 1$. As it is shown, the cycles are not closed. This is because of initial conditions. Chaotic systems such as Rössler system are extremely dependent on initial conditions. In Figures 2 we can see the attractor with the same values of a, b and c but with the initial condition of $x_0 = 2,\ y_0 = 2$ and $z_0 = 2$. This is a behavior we expect from a non-linear system. A linear non-chaotic system wouldn't behave differently for multiple initial conditions.

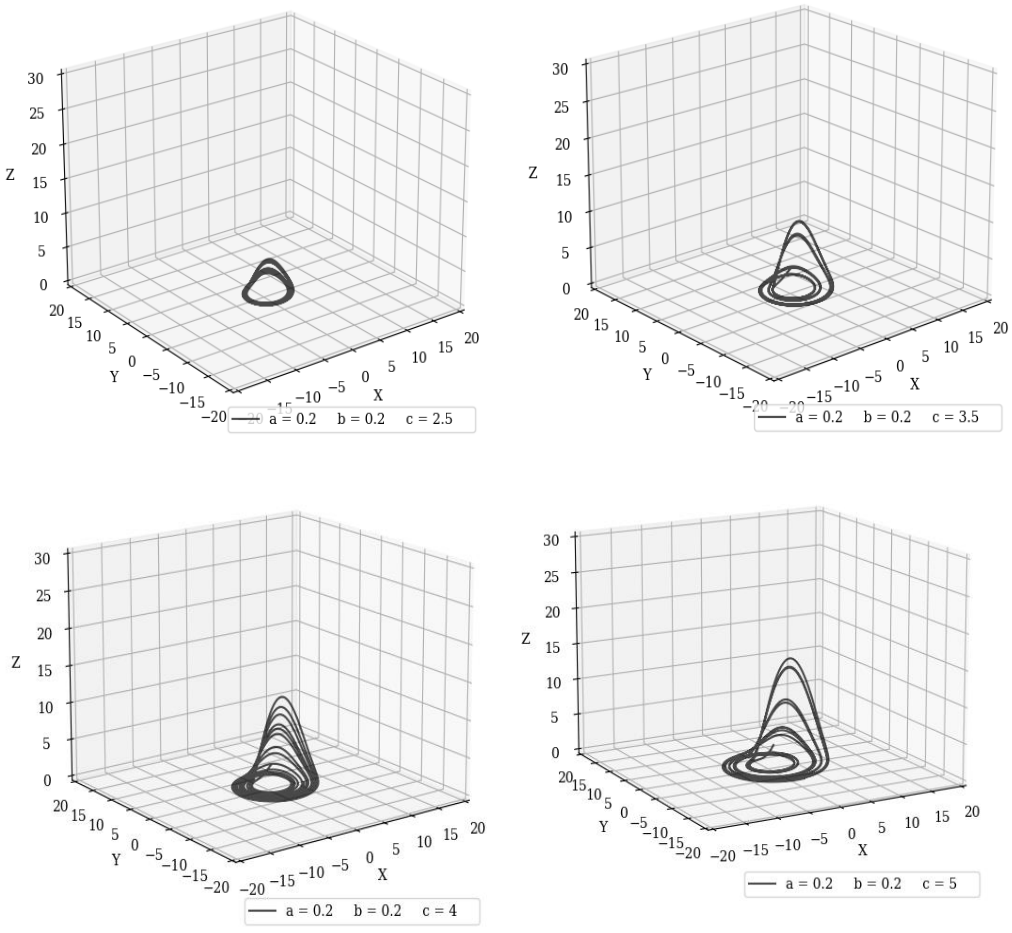

Figure 2

To notice the difference we can plot the x vs. t for two sets of different initial values (Figure 3).

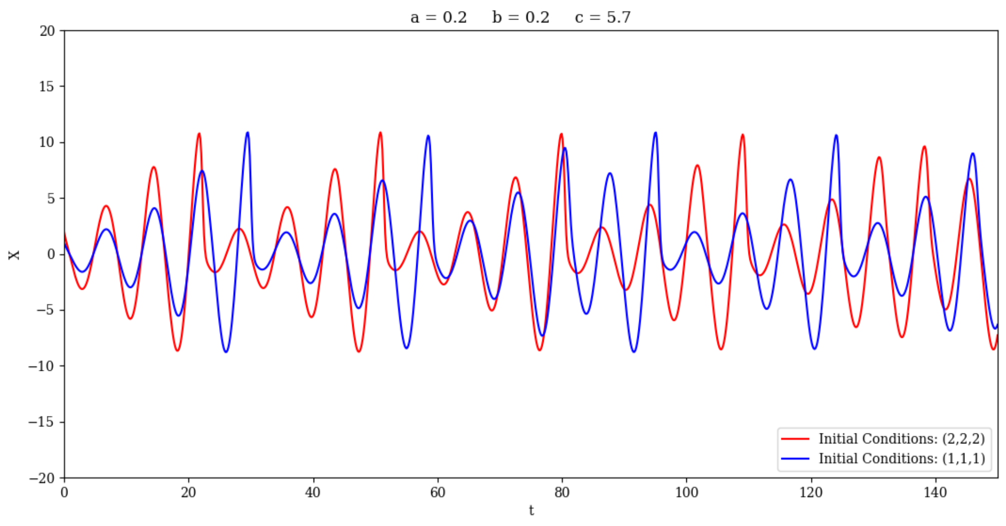

Figure 3

4. Conclusion

The Rössler attractor and the non-linear ODEs that describe the system are presenting a good example of a chaotic system. This system is highly dependent on initial conditions, which is the main feature of a chaotic system.

If the reader wants to have access to python scripts and additional details, he/she could use the following link to my GitHub repository.

Refrences

- WikiPedia
- Steven H. Strogatz, *"Nonlinear Dynamics and Chaos"*, Perseus Books, 1994