Capstone Project: Strange Attractors Simulation

Henry Hong

Abstract: The project's aim is to create plots and simulations of the following strange attractors: the Lorenz attractor, the Rössler attractor, and the Henon attractor. This was accomplished through two steps: First, I gathered data sets to plot the strange attractors using various methods. Second, I animated the plots in 2D and then in 3D.

1. Introduction

This project was inspired due to strange attractors' abundant applications in modern physics and mathematics. There is a strange attractor in every chaotic system [1]. Therefore, increasing our understanding of strange attractor behavior would further our understanding of chaotic systems, such as chaotic nonlinear oscillators. Additionally, strange attractors are the link between chaos theory and fractals.

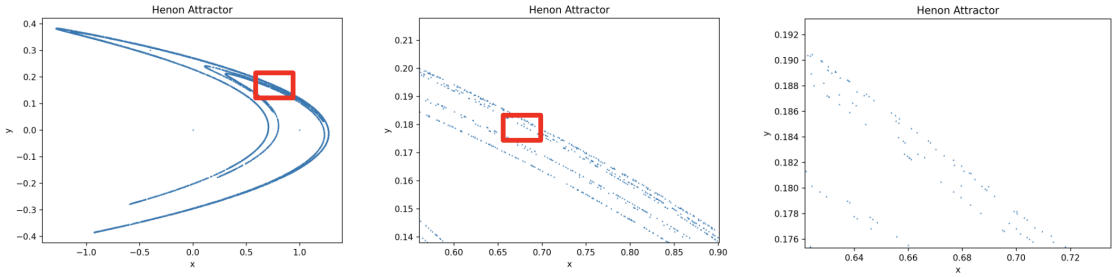


Fig. 1. Fractals in Henon Attractor

The plots show the fractal behavior in the henon attractor, as we can see a similar five-line pattern every time we zoom into the red box. The reason why the plots look faint is that there are only 10,000 data points plotted. The more data points plotted, the clearer the fractals are.

2. Calculations

2.1. Lorenz Attractor

To collect the data points necessary to plot the Lorenz attractor, one must first solve the differential equations shown in figure 2.

Fig. 2. Lorenz Attractor Differential Equations

$$dx/dt = P(y-x)$$
$$dy/dt = Rx - y - xz$$
$$dz/dt = xy - Bz$$

For the Lorenz attractor, the initial conditions are $x = 0$, $y = 1$, $z = 0$, and the constants P, R, and B, are equal to 10, 28, and 8/3 respectively. These are the values used by Edward Lorenz himself [2]. However, through experimentation, differences in the constants of up to 2 yield no noticeable changes.

In order to solve the differential equations, I used Euler's method. The process is as follows: We solve for the instantaneous slope for each dimension at each position by plugging the previous x, y, and z values into the differential equations. Next, we multiply the calculated slope by dt, which I have set as 0.01. This results in a change in x, y, or z. Adding this value to the previous x, y, or z value gives us the next value, which we can then

append to our data set. After 10,000 iterations of this process, we have a full data set, which we can then plot to get the result in Figure 3.

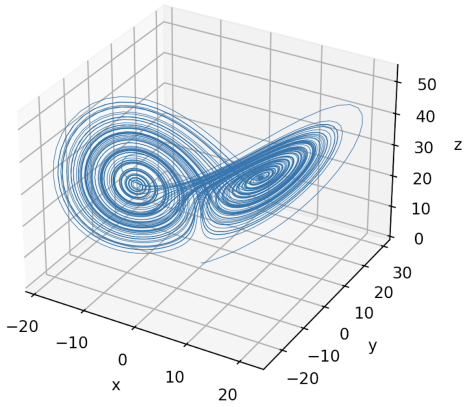


Fig. 3. Lorenz Attractor Plotted in 3-Dimensions

2.2. Rössler Attractor

Like the Lorenz attractor, one must solve the Rössler attractor's ordinary differential equations to plot it, so the process to fill the data set and plot the attractor is similar.

Fig. 4. Rossler Attractor Differential Equations

$$dx/dt = -(y+z)$$
$$dy/dt = x + Ay$$
$$dz/dt = B + xz - Cz$$

I used a new set of three differential equations (Figure 4) to plot the attractor. The constants A, B, and C are equal to 0.2, 0.2, and 5.7 respectively [3]. Using Euler's method, which is described in the previous section, we get the 3-D plot of the Rössler attractor shown below.

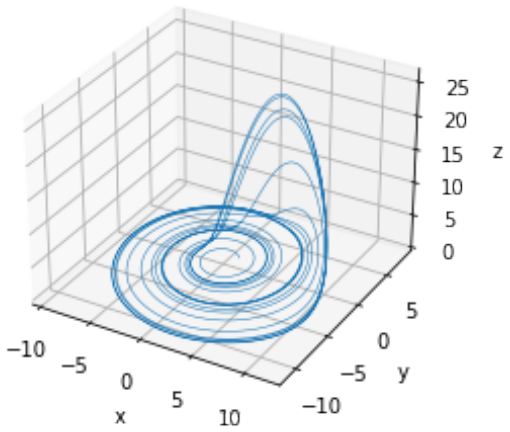


Fig. 5. Rössler Attractor Plotted in 3-Dimensions

2.3. *Henon Attractor*

Unlike the Lorenz and Rössler Equation, the equations of the Henon attractor(Figure 6) are not differential equations, so they need a method other than Euler's method in order to be solved.

Fig. 6. Henon Attractor Equations

$$x_{n+1} = (y_n + 1) - (1.4x_n^2)$$
$$y_{n+1} = 0.3x_n$$

Instead, I used recursive functions to solve the equations. Starting with the initial conditions of 0 for both x and y, we take every x and y and plug them into the equations of Figure 6 to get the next x and y values. If we repeat this process with a for loop 10,000 times, we get a data set which we can then plot to get Figure 7.

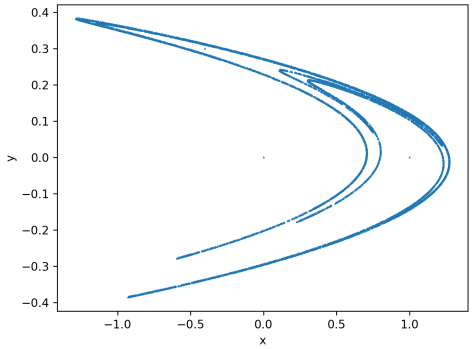


Fig. 7. Henon Attractor Plotted in 2-Dimensions

Unlike the Lorenz and Rössler attractor, the Henon attractor's path is 2-Dimensional because it only has two variables.

3. Animation

3.1. *2-Dimensional Animation*

By importing matplotlib.animation, I used the writers['ffmpeg'] feature to render the frames of my animation. Next, I created a marker, a red circle, which represented the attractor's position for each of the 10,000 data points. With a for loop, I set the red circle's position at each of the points of the data set and added each frame to my animation with grabframe. I created 2-D animations of all three attractors using this method. Figure 8 shows a snapshot of the 2-D Lorenz animation.

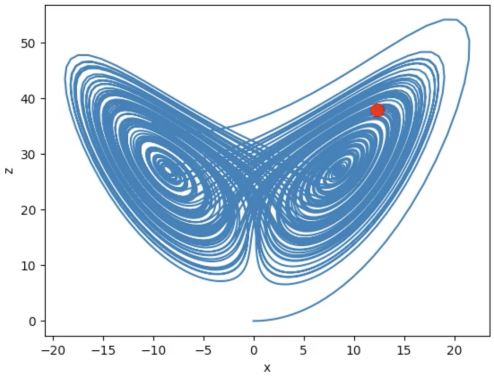


Fig. 8. 2-Dimensional animation of the Lorenz attractor

Because writers['ffmpeg'] only accepts two-dimensional data, I was unable to create three-dimensional animations with this method, which leads us to the next step.

3.2. 3-Dimensional Animation

To make a 3-Dimensional Animation, I imported FuncAnimation from the matplot.lib.animation library. Next, I defined a function that took in my three-dimensional data sets and set the animation's graph to a three-dimensional graph. Afterward, I wrapped my data set into a numpy array in preparation to be passed into FuncAnimation. I also created a starting point for animation using plt.plot and passed in my initial x, y, and z positions from the array. Finally, the function, array of my data, and the initial value into the parameters of FuncAnimation. FuncAnimation iterates through each value of the data set and generates a three-dimensional animation. I created 3-D animations for the Lorenz and Rössler attractors using this method. Figure 9 shows a snapshot of the Rössler attractor being made.

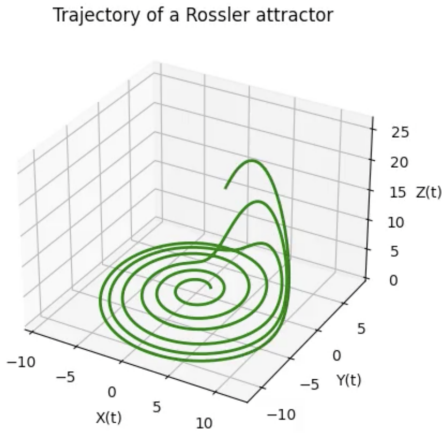


Fig. 9. 3-Dimensional animation of the Rössler attractor in progress

4. Extensions

4.1. Interactive Plots

An idea proposed by Professor Jazaeri was to make my plots interactive. To accomplish this, I imported ipympl. I also transformed each of my graphs into subplots instead of regular plots, as this is a prerequisite for ipympl. Once made interactive, plots can be dragged around, rotated, and zoomed-in or zoomed-out. By hovering your cursor over a point, ipympl also displays the coordinates of your cursor's position.

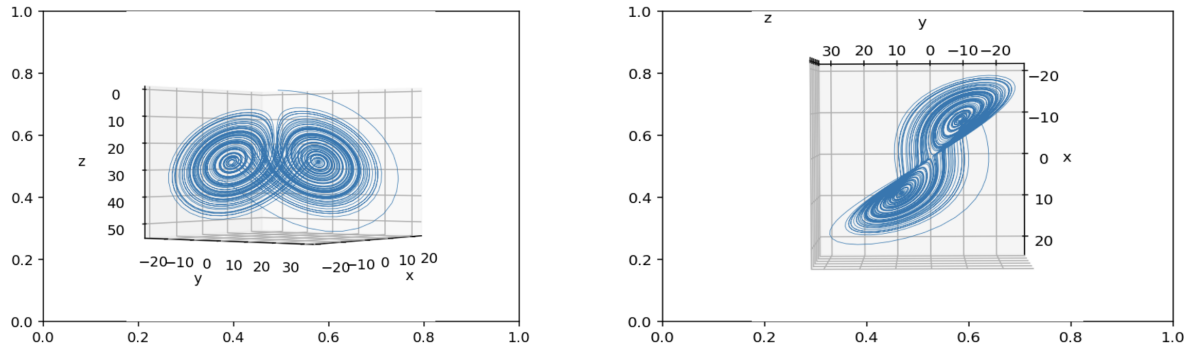


Fig. 10. Interactive Lorenz Attractor Rotations

4.2. *Comparing ODE Solving Methods*

Another extension to my project was to compare Euler's and Modified Euler's method of solving ordinary differential equations. For the Modified Euler's we take the average of both the previous point's slope as well as the next point's slope instead of just the previous slope. I get the next slope value by first creating a prediction with Euler's method. This process is called the predictor-corrector method. The average slope is then multiplied by dt to get the change in x, y, or z, and then appended to the previous value to get the next. Modified Euler's method is more accurate than Euler's method. I plotted the Lorenz attractor once using Euler's method and once using Modified Euler's method to fill its data set. The results are shown in Figure 11.

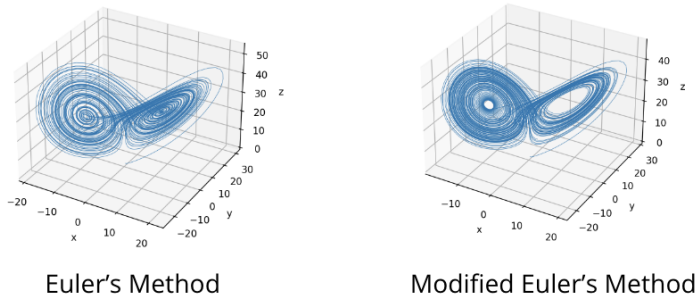
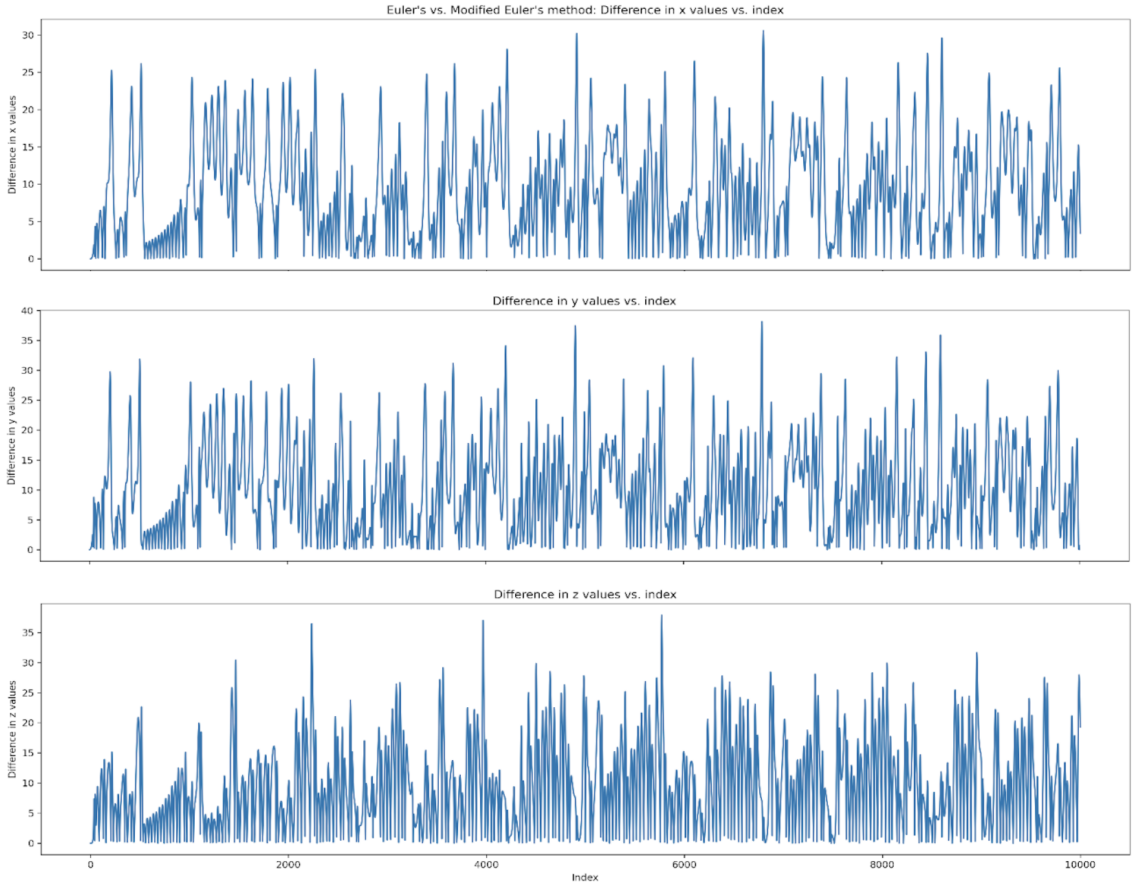


Fig. 11. Euler and Modified Euler Lorenz attractor plots

Intrigued by the noticeable differences between the two plots, I plotted the differences in values of the Euler's and Modified Euler's methods.



I believe the reason why the differences in values reach up to the high thirties is because a minor change in

values in the beginning of the plotting process ends up leading to vast changes later on, a trait unique to strange attractors. It is also interesting to note that the differences in values appear to follow the same trend across all three dimensions' values.

5. Conclusion

The final output of my project includes interactive 2-Dimensional and 3-Dimensional plots and animations of the Lorenz, Rössler, and Henon attractors. It also includes a comparison of plots solved using Euler's and Modified Euler's method. Creating and coding the algorithm to plot each of the attractors went surprisingly smooth. However, it was the animation aspect that proved difficult. I learned animation and Modified Euler's method from scratch and spent much more time debugging my code than expected. Installing and importing ipympl for interactivity also proved challenging, as it created an implementation error that was difficult to solve. However, after hours of trial and error, all of these problems were eventually resolved. Overall, this project was extremely rewarding. It served as both an excellent way to test out the skills I learned in class and as a great way to dig into and experiment with a subject on the frontier of physics research. I aim to expand this project in the near future by plotting even more complex strange attractors and experimenting with a variety of initial conditions.

References

1. "Chaos, Fractals and Dynamics." *Simon Fraser University,* https://www.sfu.ca/~rpyke/335/summary.html.
2. Faculty of Mathematics. "Chaos Theory and the Lorenz Equation: History, Analysis, and Application." *University of Waterloo,* https://links.uwaterloo.ca/pmath370w14/PMATH370/lorenz%20Latex.pdf.
3. Bradley, Larry. "Strange Attractors - Chaos & Fractals." *Space Telescope Science Institute,* https://www.stsci.edu/~lbradley/seminar/attractors.html.