CTA200 2025 Assignment 3

Zehao Peng

Question 1

For this question, I begin by defining the iteration function ziterations() in my helper Python file, utils.py. ziterations() takes a complex number c and an integer max_iter as input, and returns a boolean indicating whether the sequence diverges (True) or converges (False) within the specified number of iterations (here set to 100). The function iterates through the sequence $z_{n+1} = z_n^2 + c$, starting with $z_0 = 0$, and checks if the absolute value of z_n exceeds 2. If it does, the function returns True and the number of iterations taken to reach divergence. Otherwise, it returns False and the number of iterations taken to reach convergence. To distinguish which points converge and which points diverge, I loop through all points within the specified range $-2 < x < 2$ and $-2 < y < 2$: for those which diverge, their x-y coordinate is marked on a mask with a divergence marker. Similarly, for points which converge, their corresponding steps until convergence are recorded in another mask. To create the first image, I simply use the first mask to create a bipartition of colour. The second image is generated with colours in accordance with the second mask. The mask is a 2D array of the same size as the image created with np's meshgrid(), where each pixel corresponds to a point in the complex plane. The divergence mask is filled with 1s for points that diverge and 0s for points that converge, while the step mask is filled with the number of iterations until divergence for points that diverge and 0s for points that converge.

Question 2

I used solve_ivp to integrate Lorentz's equations. Like Question 1, I defined the actual Lorentz system in my helper python file, utils.py. This lorentz() function takes in the time t and the state vector W, and returns the derivatives of the state variables. It is written as a function that is then imported to my Jupyter Notebook, and solve_ivp was run with the specified parameters and starting conditions in the handout.

My plot of $y(t)$ matches Lorentz's exactly up to approximately 900 iterations, but begins to diverge after. This is likely due to the fact that I am using a numerical method to solve the system, which is not exact. We established that this system is very sensitive to initial conditions (chaotic), so it is reasonable to expect that small errors may accumulate over many iterations and cause the system to diverge.

The deviation of 10^{-7} in the initial conditions between W_0 and W causes an exponentially growing gap between the solutions for the two, as evident by the linear growth in log space.

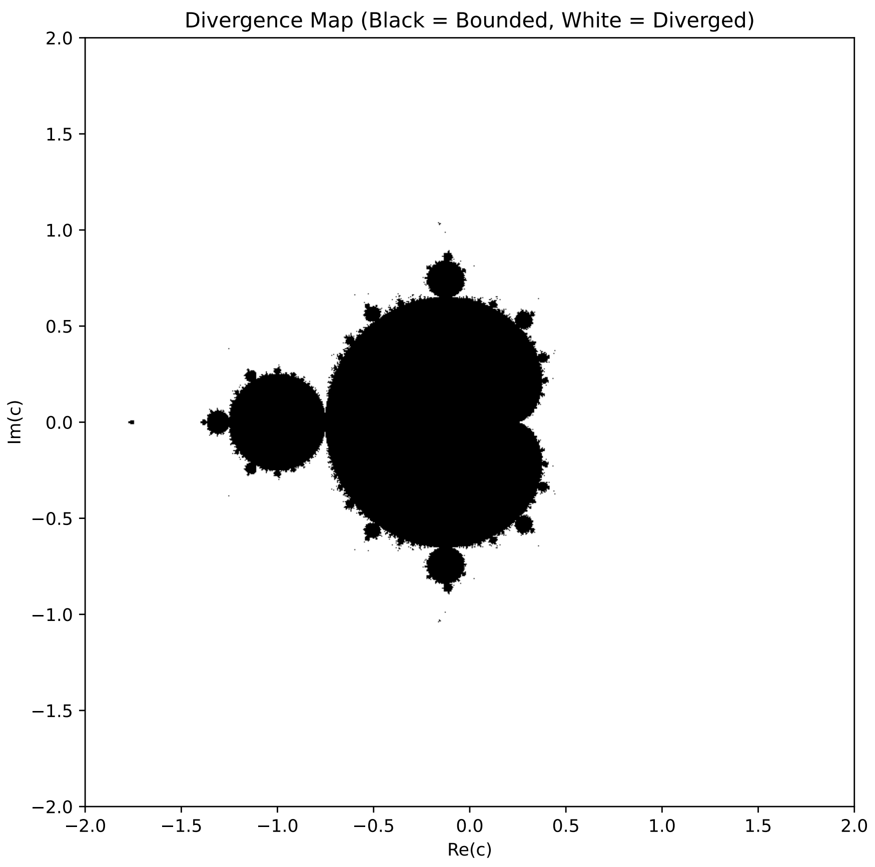

Figure 1: Divergence Map. The form assumes that of the famous Mandelbroth Set.

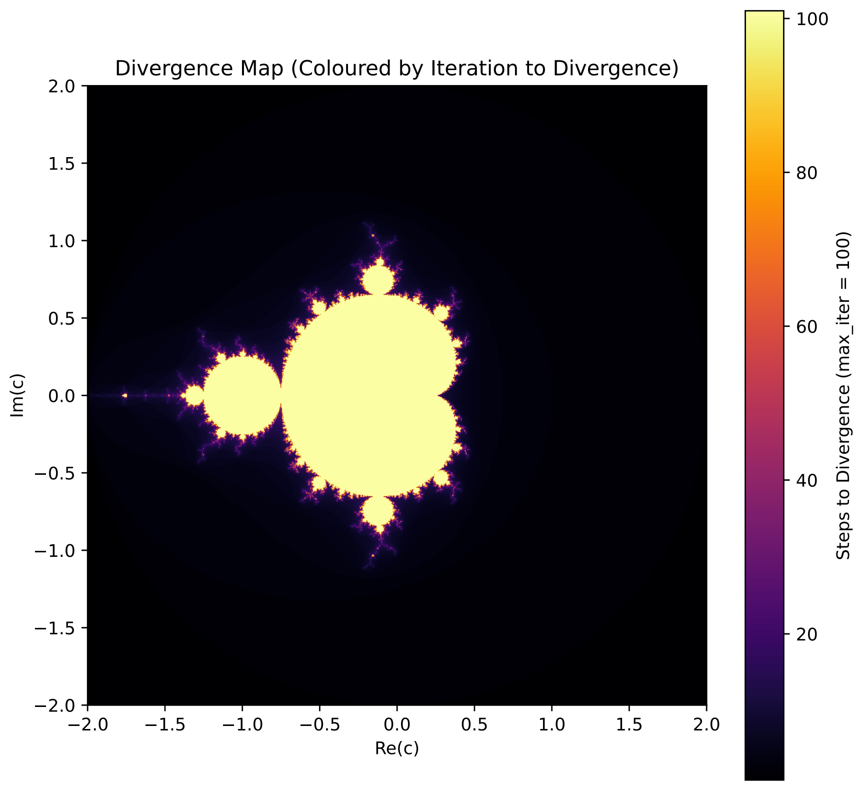

Figure 2: Divergence Step Map

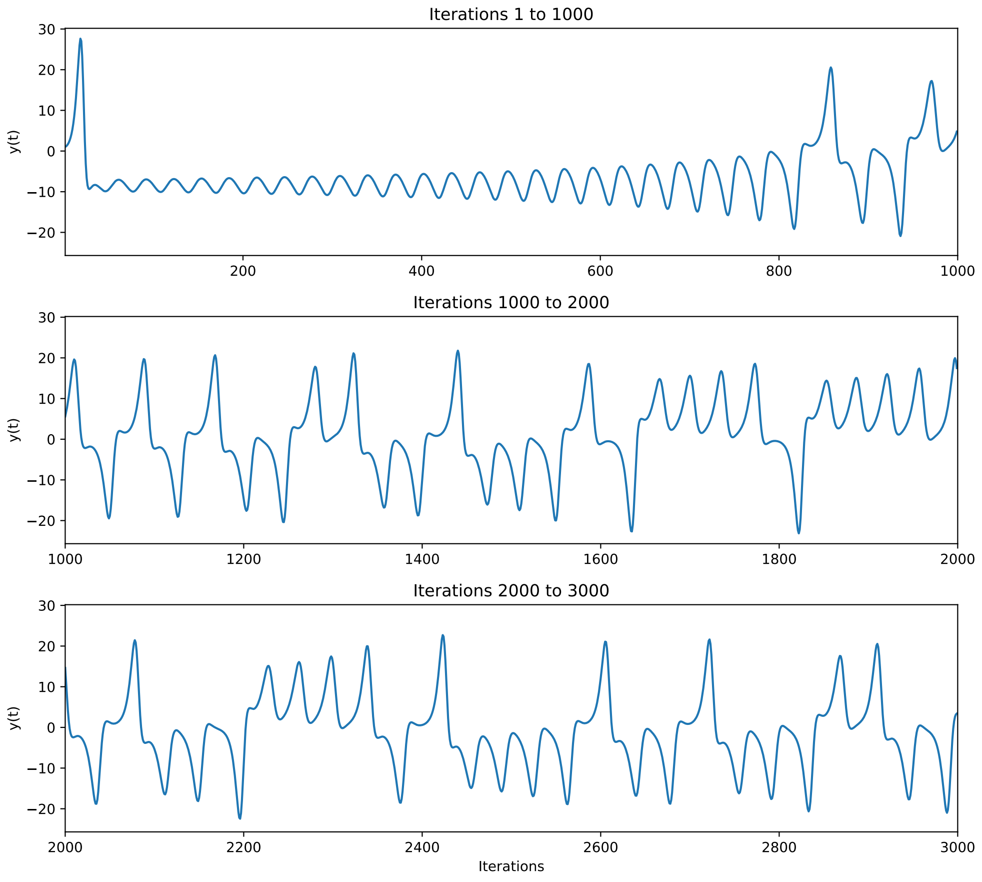

Figure 3: Numerical Solution to Lorentz's Equations. Graph of Y as a function of time for the first 1000 iterations (upper curve), second 1000 iterations (middle curve), and third 1000 iterations (lower curve). This is a reproduction of Lorentz's Figure 1.

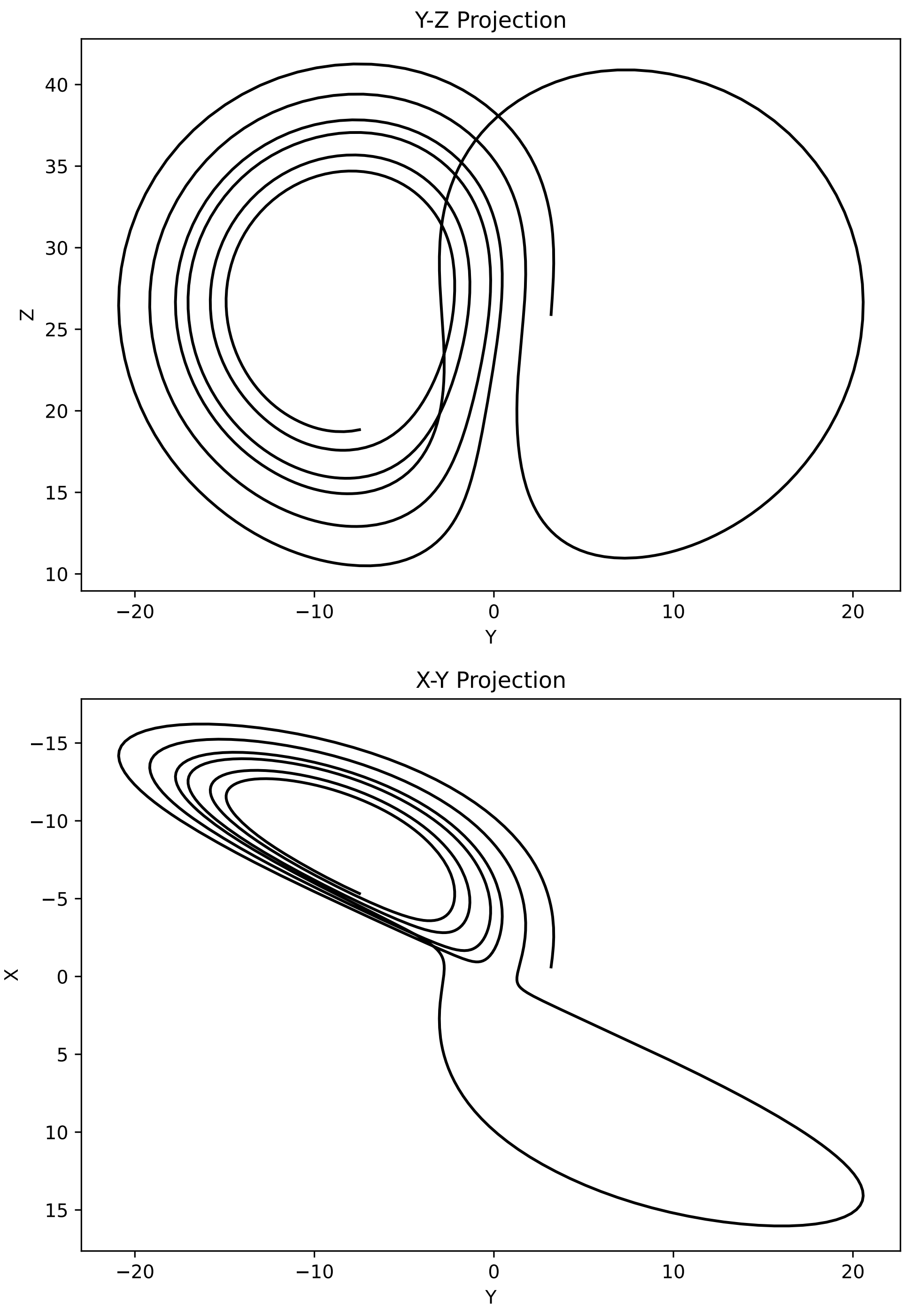

Figure 4: Numerical Solution to Lorentz's Equations. Projections on the X–Y-plane and the Y–Z-plane in phase space of the segment of the trajectory extending from iteration 1400 to iteration 1900. This is a reproduction of Lorentz's Figure 2.

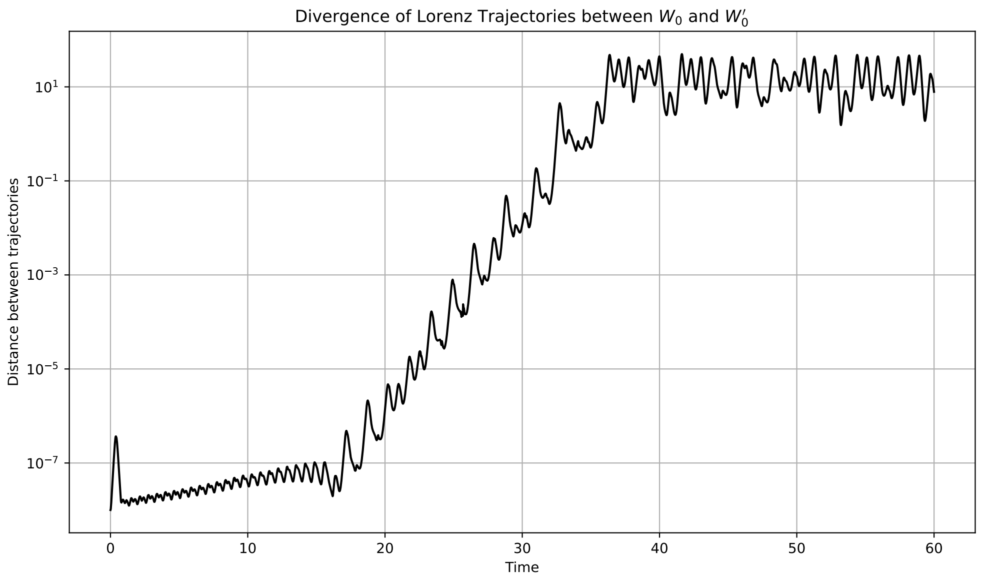

Figure 5: The distance between W' and W as a function of time, where W is generated with initial conditions $W_0 = [0, 1, 0]$ and W' generated with $W_0' = [0, 1.0000001, 0]$.