CTA200 Assignment 3 Report

Jewel Cao

May 13, 2025

1 Question 1

To designate a criteria for "divergence", I imposed the restriction that successive iterations must not exceed a given tolerance, which I set as 1e6. I created a function named 'complex_iterate' which would return the iteration number i where the $z_{i+1} - z_i > 1e6$, and 'None' if the tolerance was never exceeded. I created a contour graph indicating the number of iterations needed to exceed the tolerance (Figure 1).

Since I had produced an array with the iteration number of divergence for values in $\{c \in \mathbb{C} : -2 < Re(c), Im(c) < 2\}$, I replaced all the values of the array that were greater than 0 with a fixed constant of 10. Then, all the values of 10 corresponded to points that "diverged". I plotted this on another contour graph (Figure 2).

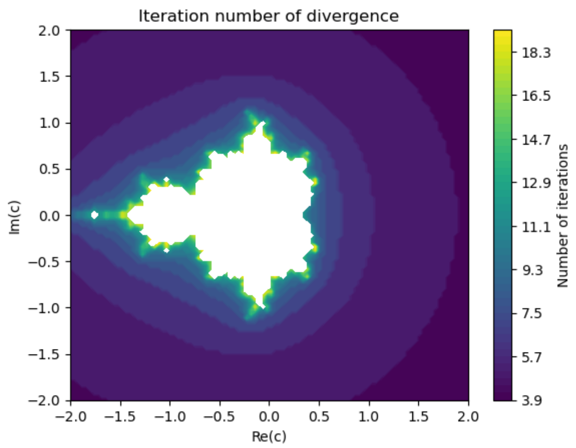

Figure 1: Contour graph indicating the number of iterations of $z_{n+1} = z_n^2 + c$ required for the difference between successive iterations to exceed the given tolerance of 1e6, for values of c in the set $\{c \in \mathbb{C} : -2 < Re(c), Im(c) < 2\}$. White indicates that the tolerance was not exceeded, and the given point converges.

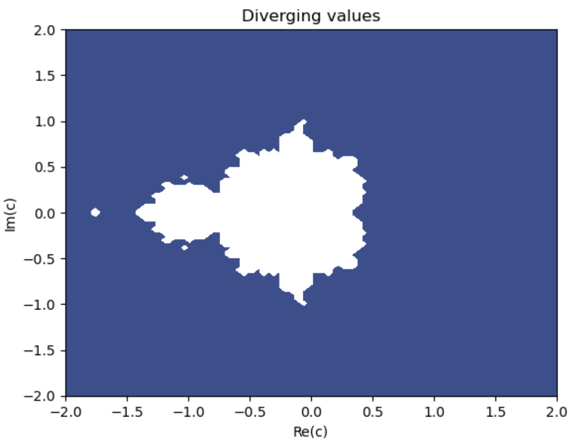

Figure 2: Contour graph indicating whether the iterative sequence $z_{n+1} = z_n^2 + c$ converges for given value of c, with c in the set $\{c \in \mathbb{C} : -2 < Re(c), Im(c) < 2\}$. White indicates that the given point converges, while blue indicates that the point diverges.

2 Question 2

I created a function that returned the right hand side of the Lorenz ODEs, and used solve_ivp for this question. I reproduced Lorenz' Figures 1 and 2 (Figures 3, 4, and 5) by closely following the process described in the assignment instructions.

In Figure 3, I plotted the timeseries of the Y component of the solution W over the entire time interval where I applied solve_ivp. In Figures 4 and 5, I used numpy linspace to 'smooth out' the solutions by applying more iterations (1000) in a smaller time interval.

In order to produce Figure 6, I repeated my previous steps to solve the ODE with slightly different initial conditions W_0'. I subtracted the two solutions W and W' to create a new array W_diff to represent the vector difference between the two solutions. Then I used the numpy linalg norm function on W_diff to calculate the scalar distance between the two solutions, which I plotted with the semilogy function in matplotlib.

As expected, Figures 3, 4, and 5 look similar to Lorenz' figures. However, Lorenz' figures were probably created over a slightly different time interval, since I was unable to specify the exact iteration numbers desired on solve_ivp. As a consequence, my figures are also plotted over units of time rather than iterations. Figure 6 appears linear on the semilog plot between units of time 0 to 40, at which point it plateaus. This suggests exponential growth of the small difference in initial condition between W and W' over this period, before diminishing into a linear increase in difference.

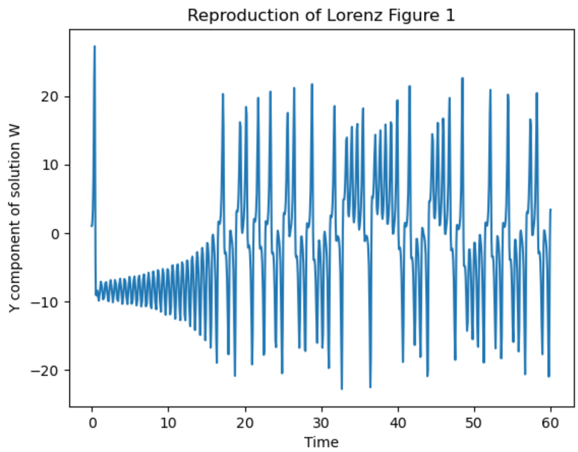

Figure 3: Y component of solution to Lorenz equations W over time, with initial parameters σ, r, b and initial conditions W_0 as described in assignment instructions.

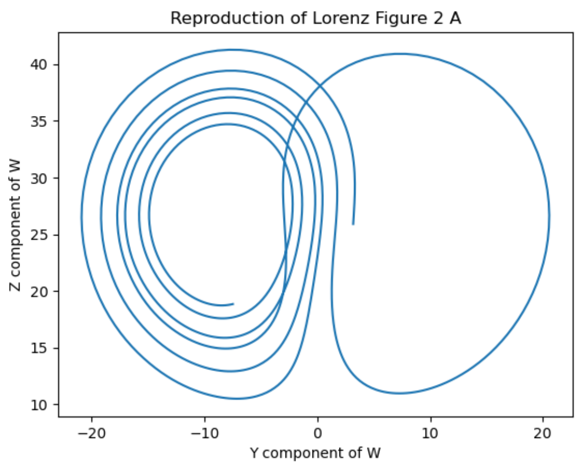

Figure 4: Phase space diagram of Y and Z components of solution to Lorenz equations W, between 14 and 19 units of time, with 1000 iterations, with initial parameters and initial conditions of Figure 2.

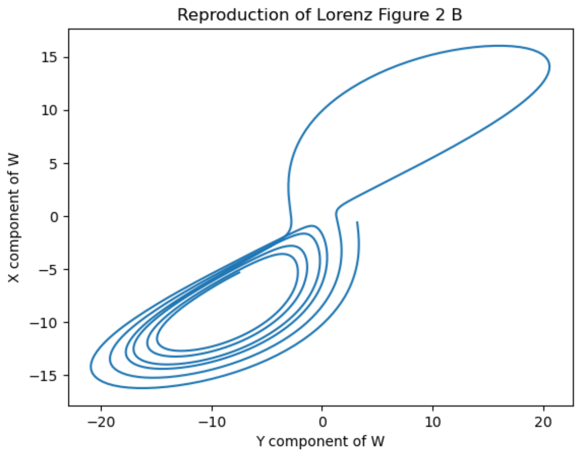

Figure 5: Phase space diagram of Y and X components of solution to Lorenz equations W, between 14 and 19 units of time, with 1000 iterations, with initial parameters and initial conditions of Figure 2.

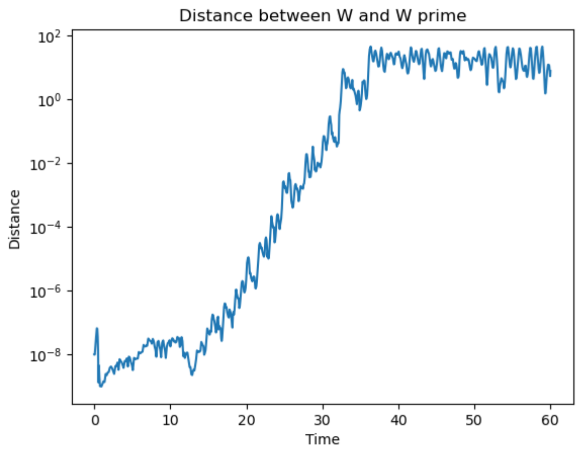

Figure 6: Norm of difference between solution W generated with initial conditions as in Figures 4 and 5 and solution W' generated with initial conditions $X = 0$, $Y = 1.e - 8$, $Z = 0$.