CTA200 2023 Assignment 3

Gowri Govindaraj

Question 1

For each point in the complex plane $c = x + iy$, with $-2 < x < 2$ and $-2 < y < 2$, set $z_0 = 0$ and iterate the equation $z_{i+1} = z_i^2 + c$. Note what happens to the z_i's: some points will remain bounded in absolute value $|z|^2 = \Re(z)^2 + \Im(z)^2$, while others will run off to infinity. Make an image in which your points c that diverge are given one color and those that stay bounded are given another. Make a second image where the points are coloured by a colourscale that indicates the iteration number at which the given point diverged.

For this question, put the code that does the iteration in a function and place this function in a separate .py file which you import in your .ipynb. Perform the plots in the notebook.

The plot is created by iterating the equation $z_{i+1} = z_i^2 + c$ for each point in the complex plane $c = x + iy$, where $-2 < x < 2$ and $-2 < y < 2$. Starting with $z_0 = 0$, the iteration is continued until the absolute value of z exceeds a certain threshold (here 2).

For points where the iteration remains bounded, the color of the pixel representing that point is set to black, indicating that the point satisfies the given condition. For points where the iteration exceeds the threshold, the color of the pixel is set to white, indicating that the point does not.

We assign colour to see the number of iterations

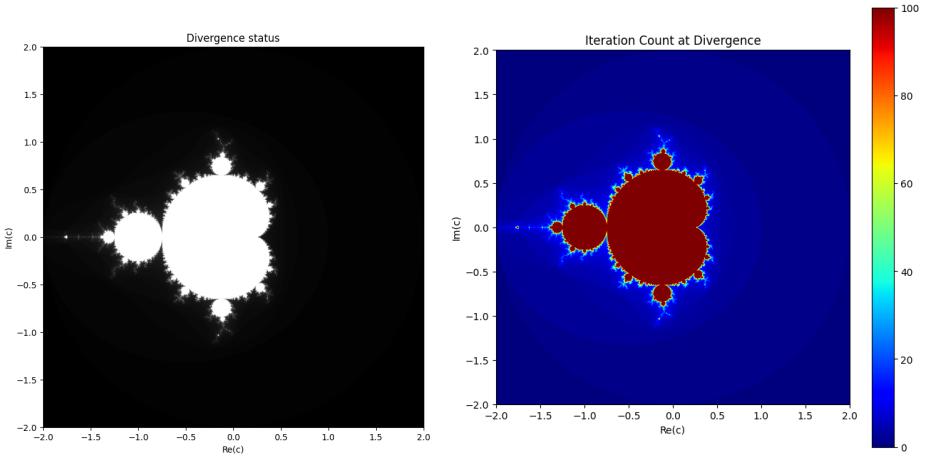

Figure 1: Plots of the divergence status and iteration count after divergence

Question 2

Introduction

One of the earliest demonstrations that deterministic physical systems could exhibit unpredictable behavior was given by Edward Lornez, a meteorologist. The original paper is linked, which is worth downloading and

looking over.

Lorenz was interested in modeling the behavior of Earth's atmosphere, i.e., a thin atmosphere (thin relative to the radius of Earth) heated from below (the air is heated by infrared radiation from the ground, or by condensing water vapor in thunder clouds, rather than by sunlight). Lorenz applies a Fourier transform to the basic equations, and truncates the number of Fourier modes, keeping only three, with amplitudes denoted by $W \equiv (X, Y, Z)$.

The equations (Lorenz' equations 25, 26, and 27) are

$$\dot{X} = -\sigma(X - Y) \quad (1)$$
$$\dot{Y} = rX - Y - XZ \quad (2)$$
$$\dot{Z} = -bZ + XY \quad (3)$$

The three dimensionless parameters are σ, the Prandtl number (the ratio of the kinematic viscosity to the thermal diffusivity), the Rayleigh number r (which depends on the vertical temperature difference between the top and bottom of the atmosphere), and b, which is a dimensionless length scale.

Note that there are non-linear terms in the second and third equations; these terms result in very complex dynamics.

Your task is to

1. code up the equations, using a function definition, with a proper docstrings (inside triple quotes)-¿ given in the code, the function lorenz_eqs is defined

2. usde ode, to integrate the equations for t=60 (in dimensionless time units). Used Lorenz' initial conditions $W_0 = [0., 1., 0.]$ and his parameter values $[\sigma, r, b] = [10., 28, 8./3.]$.

3. Reproduce Lorenz' Figure 1. Label both axes! Note that Lorenz uses $N = t/\Delta t$ to label his plots (here $\Delta t = 0.01$).

4. Reproduce Lorenz' Figure 2. You will likely have to ask for output at very closely spaced time intervals, e.g., if you use solve_ivp, you will need something like t = np.linspace(14, 19, 1000) followed by W = sol.sol(t). Again, label both axes.

5. Now find the solution using the same values of (σ, r, b), but this time with initial conditions very slightly different than W_0, say $W_0' = W_0 + [0., 1.e-8, 0] = [0., 1.00000001, 0.]$; note that adding the two lists (as indicated here) will not work, so you should google to find out how to add two lists element by element. Calculate the distance between W' and W as a function of time, and plot the result on a semilog plot (linear time, log distance). A straight line on such a plot, which is what Lorenz found, indicates exponential growth. Thus a small error in the initial condition will grow rapidly, meaning that predictions of future behavior will not be accurate.

   The first part of the code defines the Lorenz equations as a function lorenz_eq, using proper docstrings to document the input and output of the function. The function takes in the variables $[X, Y, Z]$, time t, and the parameters σ, r, and b, and returns the derivatives of the variables with respect to time.

   The second part of the code sets the values of σ, r, and b, and the initial conditions W_0. It then defines a time array t and uses odeint from scipy.integrate to solve the Lorenz equations for t=60. The solution is stored in the variable sol.

   we calculated the distance between two solutions of Lorenz equations with slightly different initial conditions, W and W'. Specifically, we used $W_0 = [0, 1, 0]$ and $W_0' = [0, 1 + \epsilon, 0]$, where $\epsilon = 10^{-8}$. We then calculated the Euclidean distance between W and W' as a function of time.

   We then plot the distance on a semilog plot, i.e., we plotted distance versus time using a logarithmic scale for the distance axis. This type of plot is useful when there is a wide range of values for one variable, in this case, the distance.

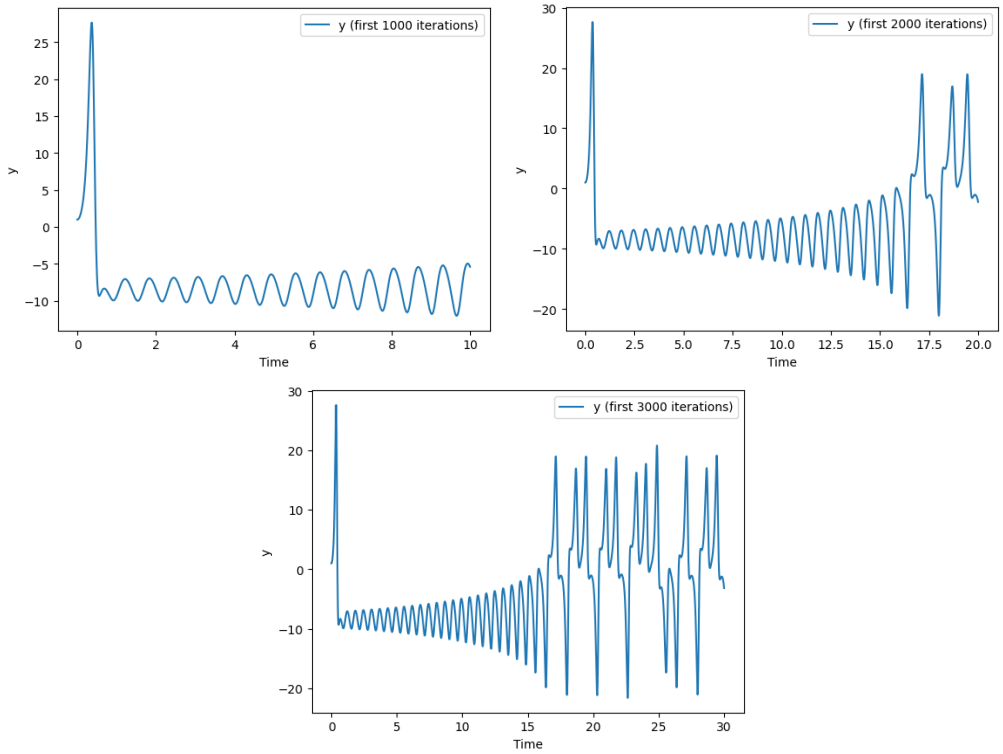

Figure 2: Plot of y vs. time for various iterations

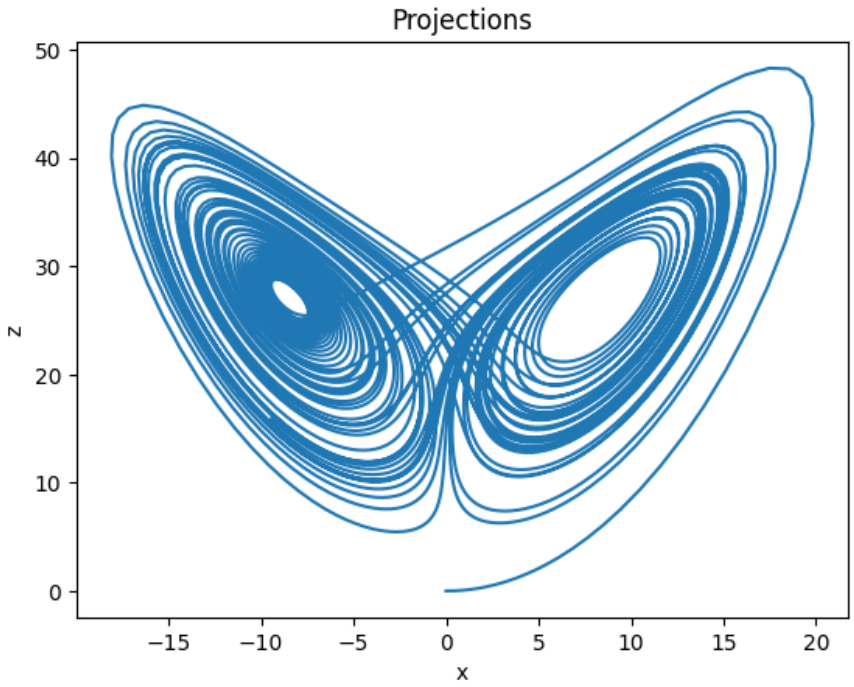

Figure 3: Solution of the convection equations

The plot shows that the distance between W and W' grows rapidly over time, indicating that small differences in initial conditions lead to vastly different outcomes over time. The plot also shows that the growth in distance is roughly exponential, as indicated by the almost-straight line on the semilog plot. This is consistent with what Lorenz found in his original paper and is now known as the butterfly effect: small differences in initial conditions can have a large impact on the long-term behavior of a system. In the case of the Lorenz equations, this means that long-term weather forecasting is impossible since small errors in initial conditions lead to large errors in predictions.

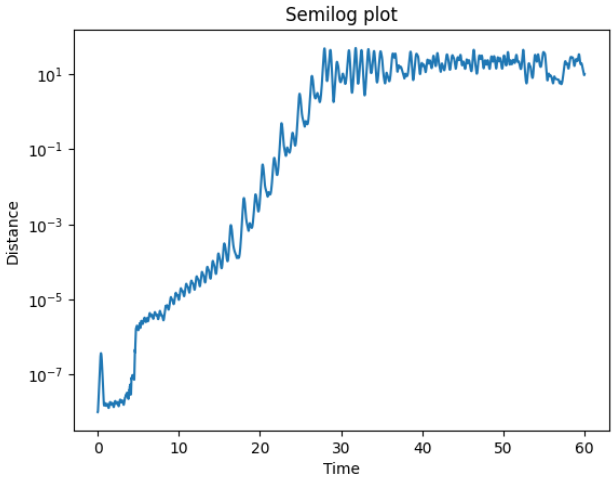

Figure 4: Semilog plot of the distance between the 2 solutions

For this question you may put all of your code in the jupyter notebook or place some of it in a separate .py file as you see fit. The only requirement is that you make sure it runs correctly in the correct order (ie restart your kernel and run each cell in order to check it works correctly before submitting).

Question 3

Writeup your results in a latex file which includes at least one plot for each of the questions as well as a description of the methods and the result. Submit the .tex file and a pdf generated from it. Yopur writeup should be 1-3 pages long, excluding figures. Save your figures as a .pdf file.

How to Submit

Submit your assignment by creating a folder called assignment_3 in your repository from Assignment #1 and put the notebook, the .py files, the `LaTex` file and the PDF output from `LaTex` as well as any other files which are necessary for me to run your code. Commit the files and push to github. No need to email me your repo again since I have the URL from assignment 1.