AM 160 - SciML:

Dante Buhl

March 5, 2025

Problem 1: Learning the Lorenz' 96 Chaotic Attractor

The lorenz' 96 system is given by the following dynamical system for $i, j, k \in [1, 8]$.

$$\frac{dX_i}{dt} = X_{i-1}\left(X_{i+1} - X_{i-2}\right) - X_i + F - \frac{hc}{b}\sum_j Y_{i,j}$$

$$\frac{dY_{i,j}}{dt} = -cbY_{i,j-1}\left(X_{i,j+2} - Y_{i,j-1}\right) - cY_{i,j} + \frac{hc}{b}X_i - \frac{he}{d}\sum_k Z_{i,j,k}$$

$$\frac{dZ_{i,j,k}}{dt} = edZ_{i,j,k-1}\left(Z_{i,j,k+1} - X_{i,j,k-2}\right) - geZ_{i,j,k} + \frac{he}{d}Y_{i,j}$$

Thus it is an 584-dimensional system (8 x variables, 64 y variables, and 512 z variables), and therfore its trajectories draw out a chaotic attractor in 584-dimensional space. Sufficient parameters to create such a system are $h = 0.5$, $c = 8$, $b = e = d = 10$, and $F = 20$. In this assignment we will be ignoring the z variables completely (including the dependence of the y variables on z), and evolving the system to see if we can learn to predict the y variables using only the x positions as an input.

a). I will preface this by saying I have no idea what the procedure went over in class was and I don't use the published code to base my code. So I will hope that I am doing what the assignment describes and proceed to respond accordingly. The first task is to train a model to predict each of the 64 y variables using only the state of the x variables. This requires at the bare minimum that we have a model that takes in 8 inputs and after some variable number of hidden layers returns an output with 64 outputs. I believe the entire context of how we train the network, what we use as a loss function, and everything else is variable and up to the student, because the assignment is very vague and doesn't describe any other constraints.

Thus I built a network and with I think 8 hidden layers and using the ReLU activation function between each layer. For training data, I used about 10-20 different trajectories of the Lorenz 96 system evolved from an RK4 integrator starting from initial conditions sampled from the the standard normal

distribution. For each timeseries, about 10 epochs were used to train the model, and as a loss function, I chose the MSELoss function and specifically took the loss of the actual prediction error $|Y_{Act} - Y_{Pred}|$ and the loss of the term in the x equation $|\sum_j Y_{i,j,Act} - \sum_j Y_{i,j,Pred}|$ scaled up by 100 (so that it is more important than actually getting the y vector correct).

$$\text{Loss} = ||Y_{i,j,\text{Act}} - Y_{i,j,\text{Pred}}||_2^2 + 100 \left|\left| \sum_j Y_{i,j,\text{Act}} - \sum_j Y_{i,j,\text{Pred}} \right|\right|_2^2$$

As shown in the output.txt file, using this method we are able to test the prediction error and find that the average error in 0.04 for Y and 0.81 for $\sum_j Y_{i,k}$. We can visualize the inaccuracy between the model and prediction in the same manner that Ashesh's sample code does. In Figure 1, we see the disparity between the prediction and actual values for $\sum_j Y_{i,j}$ for each value of i at an arbitrary timestep. The error seems to be relatively small (though some can be off by as much as a factor of 50%.

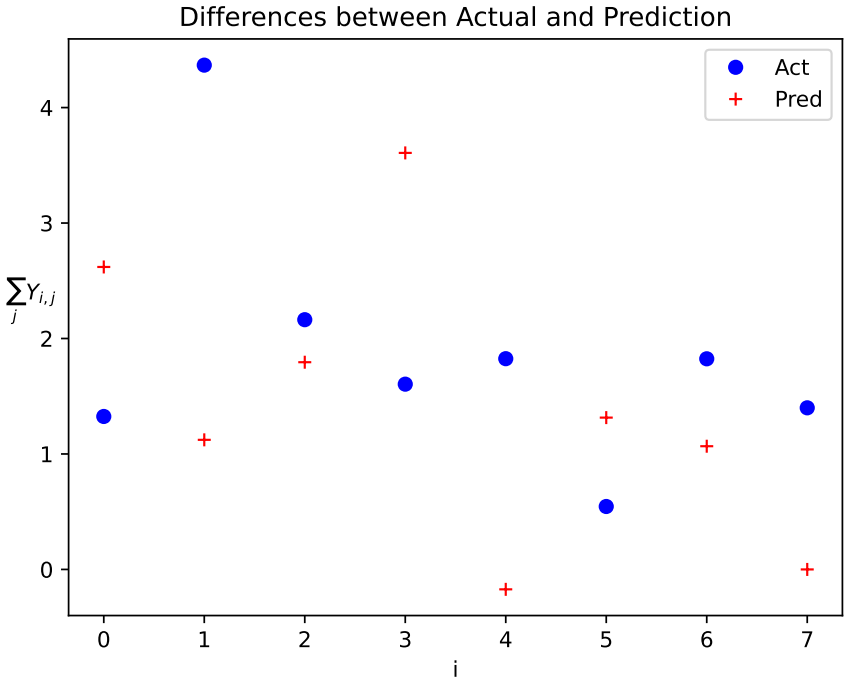

Figure 1: Plot of $\sum_j Y_{i,j}$ for each value of i chosen from an arbitrary timestep taken from a given testing timeseries.

b). With $F = 24$, I am able to get relatively low prediction error. As shown in the output.txt file, the error obtained for $Y_{i,j}$ prediction is on average 0.05... and the error obtained for $\sum_j Y_{i,j}$ is on average

1.02 . . .. After training the model on the trajectory, the average loss is on the order of 0.039 . . . and 0.002 . . . respectively.

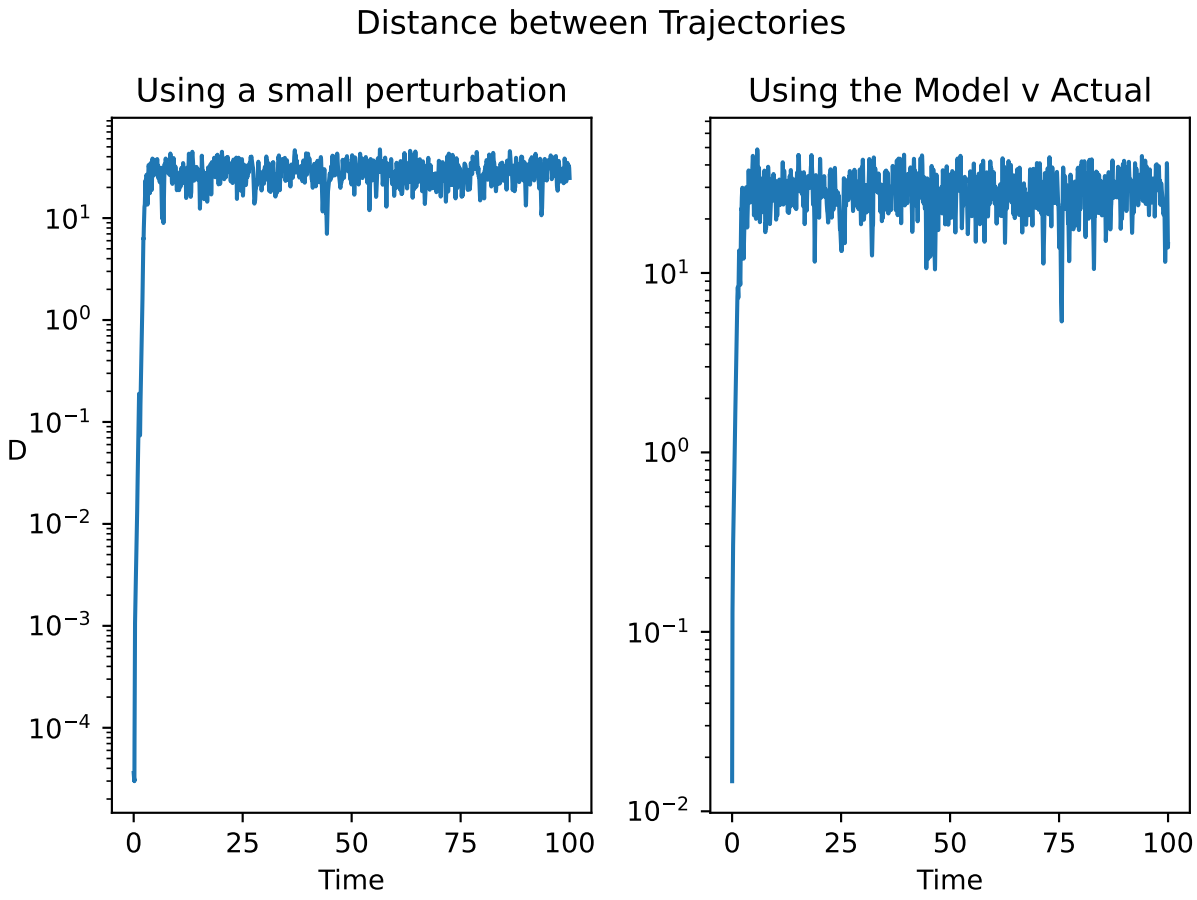

Figure 2: Natural divergence between trajectories with infinitesimal perturbation using Lorenz96 (left) and divergence between trajectories with identical IC computed with Lorenz96M and Lorenz96 (right). Note that both stabilize to $O(10)$ which indicates that the Lorenz96M system is able to recreate an attractor of the same general size (and hopefully shape), i.e. the model has successfully replicated the dynamics with a new lyapunov exponent.

c). Writing an RK4 integrator for this problem was very easy, the only change is that now the RK4 integrator evolves the Lorenz96M system where there are only 8 variables (the x variables) and furthermore, the model is introduced to make the x variables independent of y and only need itself to evolve it time. That is,

$$\frac{dX_i}{dt} = X_{i-1}\left(X_{i+1} - X_{i-2}\right) - X_i + F - \frac{hc}{b}\sum_j M_{i,j}(\boldsymbol{X})$$

Note that this system is already stabilized at this point specifically for two reasons. First and foremost,

the model is trained to primarily learn $S = \sum_j Y_{i,j}$ rather than $Y_{i,j}$ which is the pivotal point in reducing the complexity of the system. The next reason that this is stabilized is that by training on multiple trajectories which exist over long timeseries is that due to the chaotic nature of the system, the model has effectly (or rather hopefully) learned the attractor which the solution finds itself in. In order to demonstrate this, we show in Figure 2 that the model is able to predict S within a reasonable level of accuracy which simply introduces a small error that grows in the system naturally over time. The key ingredient to this being a success is that the error in the Lorenz96M system stabilizes to the same error that the Lorenz96 system has for trajectories with small perturbations. In essence, we have learned the attractor.