AMATH 575: FINAL PROJECT

ASHLEY ATKIN
EVAN KLEPITSCH
COLLIN LANDES
HARSHIL SHARMA

Applied Mathematics, University of Washington, Seattle, WA

1. Introduction

This project will focus on discovering the governing equations of dynamical systems using a data-driven approach. Many real world systems are complex and governing equations cannot be derived analytically. When analytical methods fail, it is natural to try a computational approach. However, despite having an abundance of observable and accurate data, we still struggle to determine the governing equations behind complex dynamics observed in fields such as climatology, epidemiology, and finance, among others. In this project, we will explore a novel approach called SINDy (Sparse Identification of Nonlinear Dynamics) which aims to discover governing equations from data.

SINDy has already led to advances in understanding the dynamics at play in complex physical systems like those in neuroscience. In particular, a recent article showed that SINDy was valuable in helping to uncover the dynamics at work behind perceptual decision-making [6]. The model was able to predict both choice accuracy and decision times when utilized with a multi-trial approach. The authors conclude that, "as the reaction-time task in the decision-making models is akin to first-passage-time problems, wherein the systems' dynamics may be only partially observable, our results show that the application of SINDy may be extended to similar problems in other fields, such as engineering, physics, mathematics, and biology" (Lenfesty, Bhattacharyya, Wong-Lin 2025, p.14). While the discussion in this project is not strictly dealing with a first-passage-time-problem, there will be discussion of a system with chaotic dynamics and bifurcation points and attempting to recreate the known behavior near those points.

Our paper has the following goals:

Date: June 12, 2025.

(1) Reproduce some of the results in the paper "Data-driven discovery of coordinates and governing equations" [4].
(2) Identify scenarios where the SINDy approach fails, and explore solutions (via normal forms, coordinate transformations, etc.) to address those failures.
(3) Use SINDy to predict center manifolds.

1.1. **SINDy Algorithm.** The SINDy approach considers dynamical systems of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{1}$$

where the vector $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state of the system at time t. SINDy leverages the fact that the function f usually consists of only a few relevant terms that define the dynamics, which make the governing equations "sparse" in the space of high-dimensional nonlinear functions [3]. When viewed this way, solving for the governing equations can be reduced to a regression problem where the objective is to express the dynamical system (Eq. 1) as a product of two matrices:

(1) Matrix of nonlinear candidate functions ($\boldsymbol{\Theta}$)
(2) Coefficient matrix which determines which nonlinearities are active ($\boldsymbol{\Xi}$)

$$\underbrace{\begin{bmatrix} | & | & | \\ \dot{x} & \dot{y} & \dot{z} \\ | & | & | \end{bmatrix}}_{\dot{\mathbf{X}}} = \underbrace{\begin{bmatrix} | & | & | & | & | & | & | & | & | \\ 1 & x & y & z & x^2 & xy & xz & y^2 & \dots \\ | & | & | & | & | & | & | & | & | \end{bmatrix}}_{\boldsymbol{\Theta}} \underbrace{\begin{bmatrix} | & | & | \\ \xi_1 & \xi_2 & \xi_3 \\ | & | & | \end{bmatrix}}_{\boldsymbol{\Xi}} \tag{2}$$

The SINDy approach uses a least-squares algorithm to find the coefficient matrix $\boldsymbol{\Xi}$ in a way which guarantees that each ξ_j is sparse and only a few columns of $\boldsymbol{\Theta}$ are selected [4]. In our paper we used the `pysindy` Python package [1] to run the SINDy algorithm.

1.2. **Autoencoders.** For strongly nonlinear dynamical systems, the naive library of candidate functions shown in Eq. 2 may not accurately define the dynamics. In these cases, it is common to use deep neural networks to perform a coordinate transformation on the input data. Such a coordinate transformation is done by an autoencoder, a special form of unsupervised feedforward neural network which learns to reduce input dimensionality to a "latent space" by having to reconstruct the input back up from its latent representation. For a particular dynamical system,

an autoencoder will learn an invertible coordinate tranformation which reduces the dimensionality of the dynamics. A related field of study is Koopman analysis. A Koopman operator has eigenfunctions which globally linearize the dynamics. Deep learning can be used to discover Koopman eigenfunctions from data, which provides one possible method for developing autoencoders [7].

2. Reproduction of results

In this section we present our attempt to reproduce the results in the section titled "Chaotic Lorenz System" in the paper "Data-driven discovery of coordinates and governing equations" [4]. We used the following method:

(1) Solve the Lorenz system numerically using `scipy.integrate.solve_ivp`.
(2) Inject random noise into the state space using a standard normal distribution.
(3) Create `pysindy` model using the noisy data.

Figure 2 shows that we achieved a qualitatively accurate solution to the Lorenz system using the SINDy model. Figure 3 shows the error between the actual and modeled trajectories. And Figure 1 shows that we achieved a numerical result similar to the result in the paper. There are some subtle differences between our results and the results from the research article, which can be primarily attributed to using a different noise profile as well as implementing the pysindy library.

In Figure 3 it is notable that the SINDy predicted trajectories very closely match the true trajectories for the first half of the simulation. The predicted trajectories are on top of the true trajectories for a significant time before they begin to diverge. Even after diverging, the error is primarily a difference in phase since the simulated trajectories maintain the true qualitative shape for the entirety of the simulation. This demonstrates the remarkable fact that systems with complex dynamics (and even chaos) can be modeled using a sparse set of nonlinear candidate functions. Determining which nonlinearities are active is not an easy task, but this shows that SINDy is an invaluable tool in systems modeling because it has the capability to simplify complex dynamics with high accuracy.

$$\begin{cases} \dot{x} = -10x + 10y \\ \dot{y} = 28x - y - xz \\ \dot{z} = xy - \frac{8}{3}z \end{cases} \qquad \begin{cases} \dot{x} = -10x + 10y \\ \dot{y} = 27.7x - 0.9y - 5.5xz \\ \dot{z} = 5.5xy - 2.7z \end{cases} \qquad \begin{cases} \dot{x} = -10x + 10y \\ \dot{y} = 27.6x - 0.9y - xz \\ \dot{z} = xy - 2.7z \end{cases}$$

Figure 1. (left) The true Lorenz system with $\rho = 28, \sigma = 10, \beta = 8/3$; (center) the results from [4]; (right) our results.

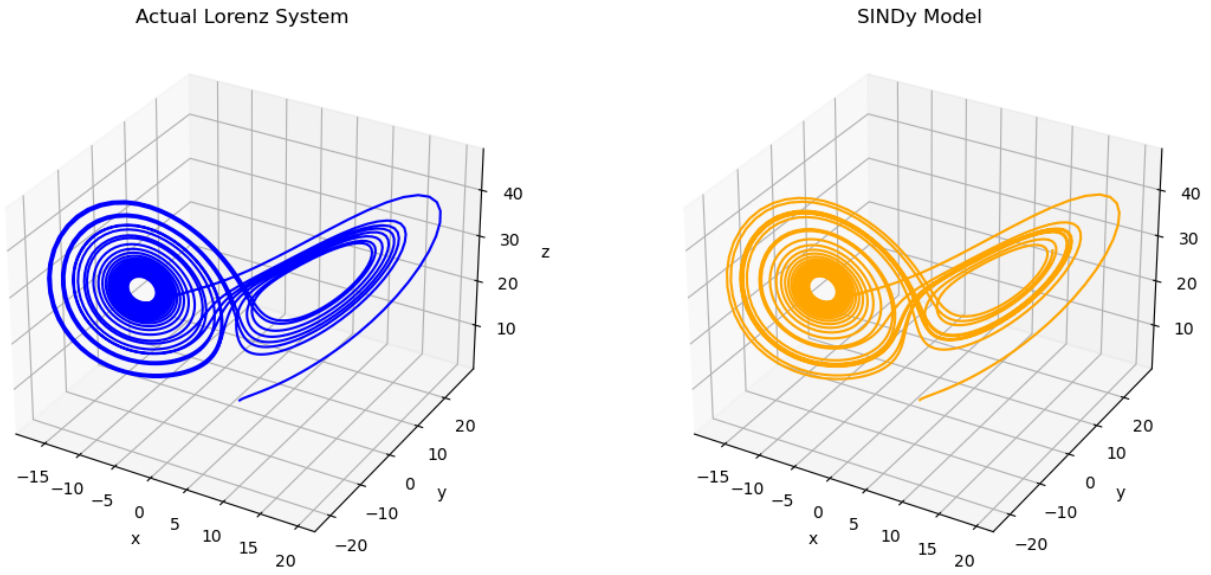


FIGURE 2. The actual solution to the Lorenz system, and our SINDy model derived from noisy data.

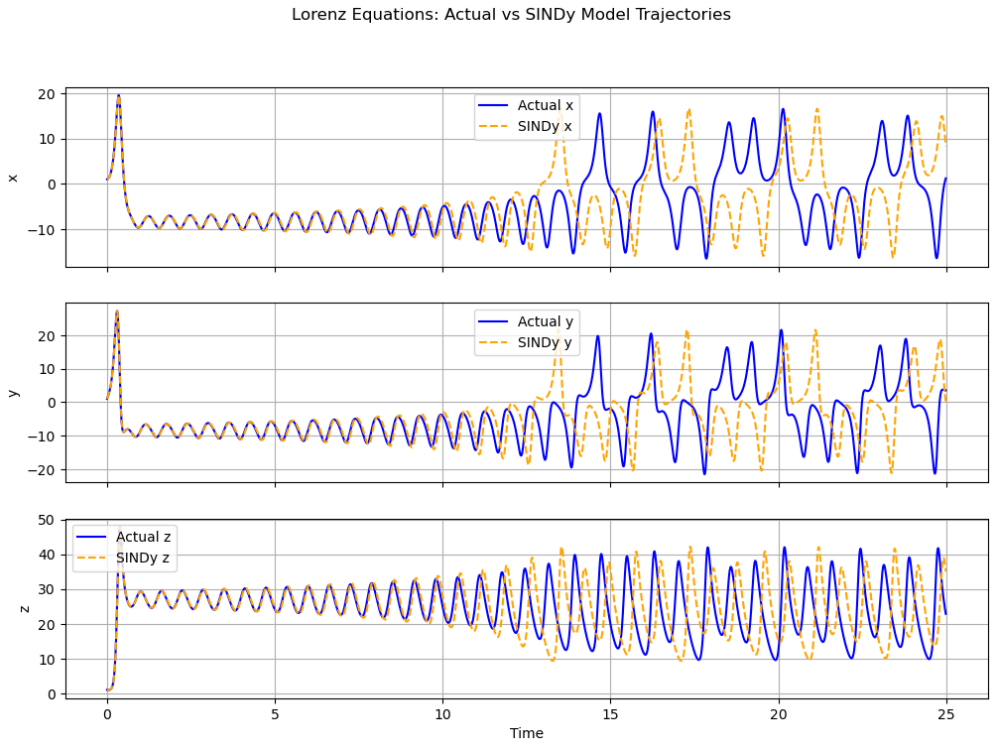


FIGURE 3. Lorenz equations modeling error: actuals vs. SINDy predicted.

We also experimented with increasing the noise profile for the SINDy model of the Lorenz system to capture qualitative as well as numerical differences with a noisier dataset. The increased noise profile plots can be found in Figures 16 and 17. From these plots, we can deduce that the SINDy

model produces a less accurate approximation for the Lorenz system. Specifically, when calculating the mean-squared-error (MSE) for each variable the original model produces a total MSE value of ≈ 85.4 versus a value of ≈ 95.0 for the noisier model, this information is included in our Jupyter notebooks located in our GitHub data repository cited in Appendix B. Code.

For further modeling purposes, we used SINDy to model the logistic equation and the Rössler attractor. We found similar success modeling these systems using the same approach as above. Please refer to Figures 8, 9, 10, 11, 12, 13, 14, and 15 in the Appendix.

3. Novel Results

3.1. **Bifurcations and normal forms.** The Lorenz system has a bifurcation point at $(x, y, z) = (0, 0, 0)$ and $\rho = 1$. We attempted to create a SINDy model near this point and failed. Put simply, simulating the Lorenz system near $\rho = 1$ does not work. The trajectories blow up and result in overflow.

Normal form theory attempts to simplify dynamical systems near specific points by capturing only the essential behavior (i.e., the terms which dominate and define the qualitative behavior). Hence, a system's normal form is less complicated to work with than the original system itself. Since modeling the Lorenz system (directly) near the bifurcation point did not work, we explored using normal forms to model the system near the bifurcation point in the hopes of achieving better results.

3.1.1. *Using the analytical normal form.* The normal form of the Lorenz equations near the origin is given by Eq. 3.

$$\begin{cases} \dot{x} = \mu x - \omega y + \alpha xz \\ \dot{y} = \omega x + \mu y + \alpha yz \\ \dot{z} = -2\beta z + b(x^2 + y^2) \end{cases} \quad \text{where} \quad \begin{aligned} \mu &= \rho - 1 \\ \omega &= \sqrt{\sigma} \\ \alpha &= \tfrac{\sigma+1}{2\sigma} \\ b &= \tfrac{1}{2} \end{aligned} \tag{3}$$

We followed the same procedure as above (solve the normal form using `solve_ivp`, inject noise, then create a SINDy model), and found that the results are highly dependent on the noise which is injected into the data. Some random noise profiles resulted in a successful SINDy model near the bifurcation point, and other random noise profiles resulted in qualitatively incorrect models. A few

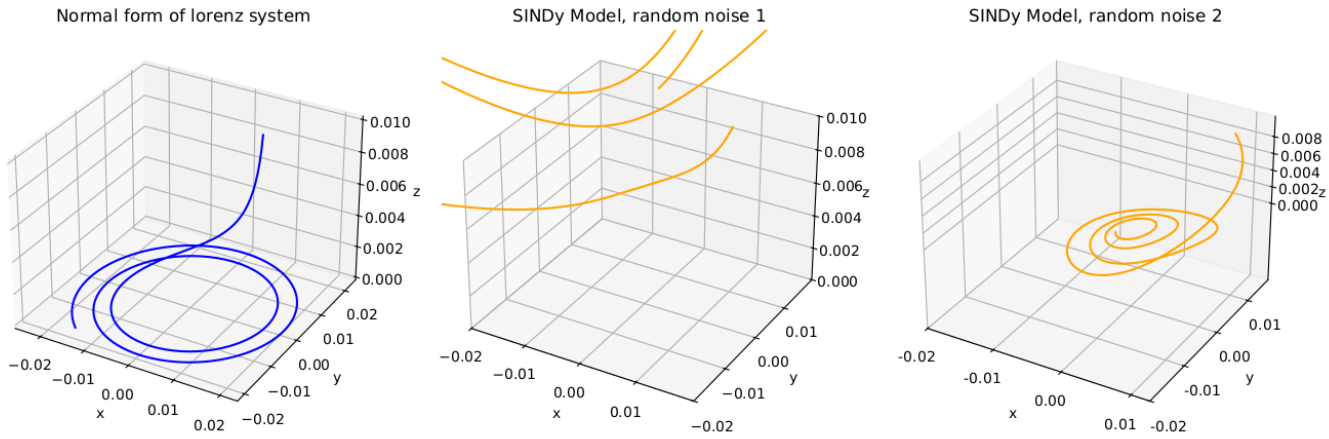


FIGURE 4. Simulation of the Lorenz system near the bifurcation point $\rho = 1$, with initial condition $(0.01, 0.01, 0, 01)$. This simulation used pysindy and an analytically derived normal form.

noise profiles resulted in a similar problem as before - trajectories blowing up and computational overflow. This is illustrated in Figure 4. The plot on the left side shows the true normal form near the origin and $\rho = 1$. The plot in the center shows the SINDy model for one particular noise profile which is qualitatively incorrect. The plot on the right side shows the SINDy model for a different noise profile which resulted in a model that has some similarities to the true normal form.

An implementation detail that we encountered when creating different noise profiles is that the noise generated by `numpy.random.standard_normal` is different when the code is run on different computers because the random number generator uses entropy from the computer to seed the initial value. This made it impossible to re-run the code and produce consistent results. The solution was to explicitly set the seed of the random number generator, which guarantees that the random noise will be the same between re-runs, and between computers. While this is not desirable in all research scenarios, it made it easier to produce consistent results for this project.

Our conclusion is that SINDy models do not consistently represent the Lorenz system near the bifurcation point, even when using the normal form. Additional methods are needed to model the system near these critical points.

3.1.2. *Using SINDy + autoencoders to compute the normal form.* The next step in modeling the Lorenz system near the bifurcation point is to combine autoencoders with SINDy. An example autoencoder implementation is provided by Champion, et al. in Github [2]. We generated the

Lorenz system data and fed it into this example autoencoder to train a neural network. The network learned a coordinate transformation suitable for linearizing the Lorenz system.

Then, we created a SINDy model of the Lorenz system using the transformed coordinates, and simulated this model near the bifurcation point $\rho = 1$. We used the same initial condition as in the previous section so that the results may be compared. The results are shown in Figure 5.

The plot on the left side of Figure 5 shows the SINDy model of the Lorenz system near the bifurcation point *without* coordinate transformations. (Note that here we used the implementation of SINDy provided by Champion, et al. [4] instead of `pysindy`). Unlike `pysindy`, the simulation did not blow up, but it is highly inaccurate. The result is a roughly linear plot that takes on small values near zero. It does not resemble the results that we simulated using the normal form in the previous section.

The plot in the center of Figure 5 shows the SINDy model that was simulated with the transformed coordinates from the autoencoder. The x, y, and z values are larger than the values in Figure 4 due to the coordinate transformation. Qualitatively we observe some oscillations which is a good sign because it is starting to bear resemblance to the true normal form.

Finally, the plot on the right side of Figure 5 shows the SINDy model after transforming it back into the original coordinate system. There is significant numerical error in the simulation with coordinate transformations because the x, y, and z values differ from the true normal form. However, the shape is qualitatively correct. This result demonstrates the power of combining machine learning with SINDy modeling. The coordinate transformation learned by the autoencoder enabled modeling the Lorenz system near the bifurcation point. Achieving such a model was not possible using other, more direct approaches.

This result is in agreement with the SINDy autoencoder results discussed by Champion, et al. in [4]. In their paper, they state "the resulting SINDy models produce dynamics that are qualitatively similar to the true trajectories," however, the coefficients of their discovered models have a difference in magnitude when compared to the coefficients in the true system. Similarly, our results were qualitatively similar to the true trajectory near the bifurcation point, but with significant differences in magnitude.

3.1.3. *Generality of autoencoders and other methods.* Adding the autoencoder to the SINDy model was a useful tool which made a difficult modeling situation less so. However, the drawbacks of autoencoders are worth discussing. In this case, the autoencoder was trained with a very limited

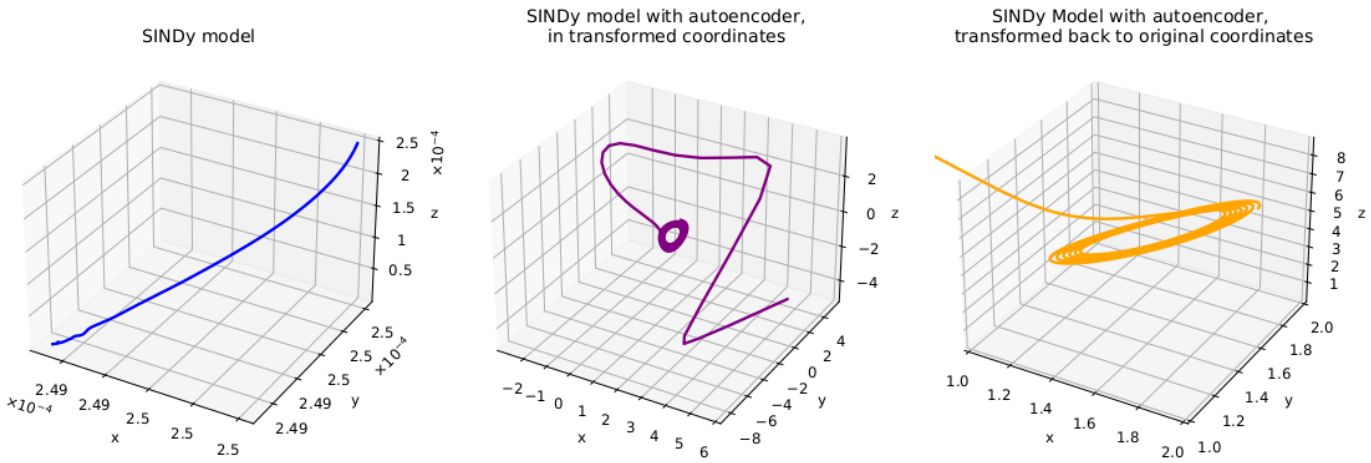


FIGURE 5. Simulation of the Lorenz system near the bifurcation point $\rho = 1$, with initial condition $(0.01, 0.01, 0, 01)$. This simulation used the autoencoder implementation provided by Champion, et al. in [4].

dataset. The training data consisted of the state space for the Lorenz system. As mentioned in [4], the resulting neural network is not interpretable or generalizable. If we wanted to use the autoencoder to learn a coordinate transformation for a different system, then it will require retraining with new data.

Another alternative to autoencoders and Neural Networks is a framework of SINDy paired with model predictive control (MPC) discussed in a separate paper also by Brunton & Kutz [5]. MPC is a technique that helps to control strongly non-linear systems with restraints. In their study, the authors showed that SINDy-MPC performed well in identifying models quickly with limited data. This is much more useful for meeting real-time identification and prediction needs such as optimizing drug therapy for patients, an application discussed in the article. In reference to recent advances in sparse model identification they state: "these innovations suggest a shift from the perspective of big data to the control-oriented perspective of smart data" (Kaiser, Kutz & Brunson 2018, p.21).

3.2. **Using SINDy to derive the center manifold.** When $\rho = 1$, the Jacobian of the Lorenz system has a zero eigenvalue and thus the system has a center manifold. In Problem Set 2 of the course, we calculated the center manifold up to second-order terms to be:

$$\begin{aligned} y &= x + O(x^3) \\ z &= \tfrac{3}{8}x^2 + O(x^3) \end{aligned} \tag{4}$$

We attempted to see if we could recover these equations or something close to them using SINDy. When the equations for the manifold are plugged back into the original Lorenz system, the result is as follows:

$$\begin{aligned} \dot{x} &= 0 \\ \dot{y} &= -\tfrac{3}{8}x^3 \\ \dot{z} &= 0 \end{aligned} \tag{5}$$

This is what we were looking for SINDy to return. In our code, we defined the specific case of the Lorenz system and then explicitly stated the Jacobian. Then, we solved for the eigenvalues and eigenvectors. We isolated the center eigenvector and generated initial conditions along the eigenvector very close to the origin. Then we simulated trajectories from those initial conditions using SciPy's `solve ivp` and plotted the result. It was as expected: parabolic in shape and tangent to the xy-plane.

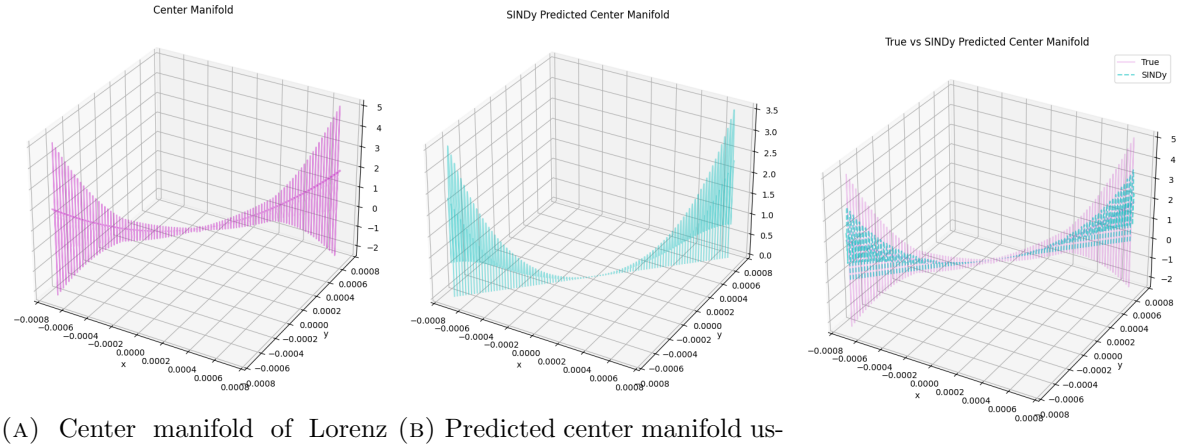
To compute the center manifold using SINDy we used the STLSQ optimizer and SINDy's polynomial library up to order 3. We then fed the model the same trajectories, printed the governing equations, and plotted the results both alone and against the expected center manifold. The results are shown in Figure 7.

The plotted trajectories given by SINDy are quite close to the expected ones for the center manifold. However, the governing equations given are not close to the computed system 6.

```
(x0)' = -0.003 x0 + 0.004 x1 + 187503.032 x0 x2 + -171539.851 x1 x2 + -1171325757.599 x0^3 + 3481409972.055 x0^2 x1 + -34485
75630.496 x0 x1^2 + 1138485394.738 x1^3
(x1)' = -0.002 x0 + 0.003 x1 + 137485796.479 x0^3 + -411104308.769 x0^2 x1 + 409753147.879 x0 x1^2 + -136134614.752 x1^3
(x2)' = -0.276 x2 + -1327.665 x0^2 + 2625.503 x0 x1 + -1297.702 x1^2
```

FIGURE 6. Resulting governing equations predicted by SINDy.

One reason for why this might be is that the trajectories are along the linearized center eigenvector direction but the center manifold is nonlinear and the trajectories quickly leave the manifold and veer to the stable directions. Another issue is that SINDy is trying to learn the full 3D dynamics of the system with the trajectories given and the center manifold is one-dimensional in this case. Also, using the polynomial library with degree 3 over a tiny data range can result in large coefficients and affect the conditioning of the problem.



(A) Center manifold of Lorenz system near r=1 (B) Predicted center manifold using SINDy (C) Combined plot

FIGURE 7. Center manifolds of Lorenz system: actual and predicted.

4. Conclusion

In conclusion, we were able to successfully reproduce the results of the "Chaotic Lorenz System" discussed in the research article [4]. SINDy proved itself capable of rendering an accurate approximation. However, as expected, the noisier the dataset, the less accurate the SINDy model. Likewise, this was apparent when using SINDy to model the Lorenz system near the bifurcation point at the origin where $\rho = 1$, even when considering normal forms. Also, when implementing SINDy with autoencoders and coordinate transformations to compute the normal form of the Lorenz system near a bifurcation point, we were able to render results that are qualitatively accurate in terms of trajectories but experienced significant difference in magnitude. Furthermore, when using SINDy to derive the center manifold of the Lorenz equations, we experienced mixed results. SINDy was able to capture close approximations of the trajectories for the center manifold, but the model for the governing equations did not align with the true system.

In general, SINDy performs well in modeling nonlinear systems with many variables. Leveraging sparse regression to model nonlinear systems can be considered a compelling addition to the field of dynamical systems. Not only can SINDy model nonlinear dynamical systems with known governing equations, its capabilities can be extended to real-world datasets in the hopes of deriving the underlying system where analytical solutions are unknown.

However, SINDy has limitations. Although a powerful modern tool, SINDy is sensitive to changes in input, as evidenced by our experiments. Moreover, when implementing SINDy with autoencoders there is a level of qualititative accuracy but a deficiency in terms of numerical accuracy, i.e., there

is a discrepancy between the predicted coefficients and the actuals. Lastly, while no experiments were performed for this particular project, it should also be noted that when modeling real-world data, it is difficult to select the appropriate basis terms from the SINDy candidate library. Without prior knowledge of the structure and behavior of a system, SINDy can produce models that are difficult to interpret.

References

[1] Pysindy. `https://pysindy.readthedocs.io/en/latest/`. [Online; accessed 5-June-2025].

[2] Sindyautoencoders. `https://github.com/kpchamp/SindyAutoencoders`. [Online; accessed 10-June-2025].

[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[4] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[5] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proc. R. Soc. A*, 2018.

[6] B. Lenfesty, S. Bhattacharyya, and K. Wong-Lin. Uncovering dynamical equations of stochastic decision models using data-driven sindy algorithm. *Neural Computation*, 37(3):569–587, 02 2025.

[7] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun*, 2018.

Appendix A. Supplementary figures

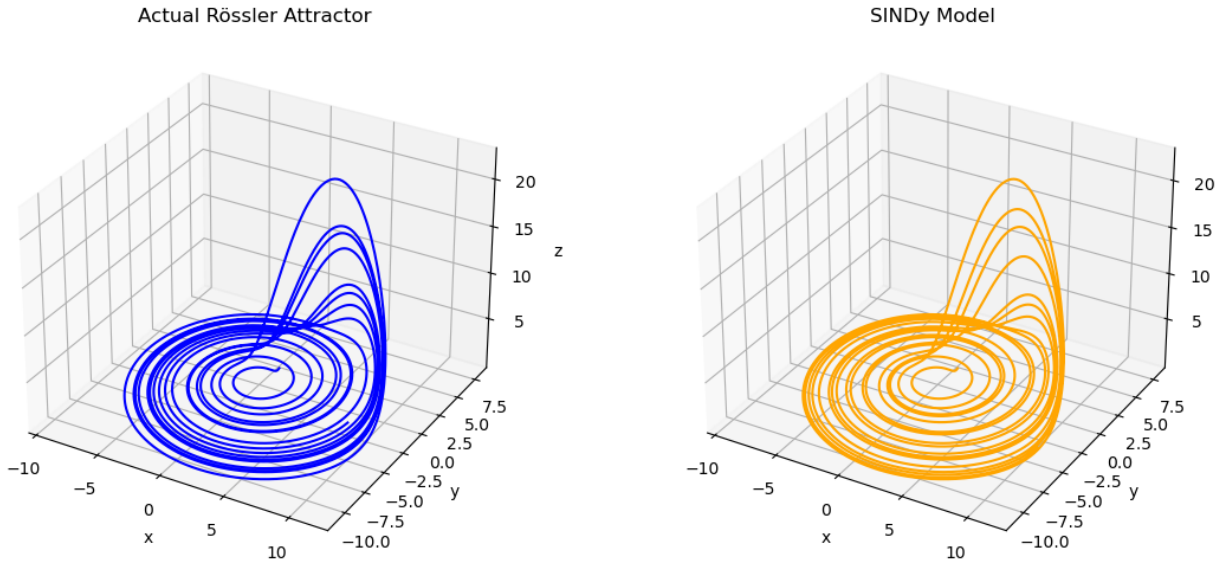


Figure 8. The actual solution to the Rössler attractor, and our SINDy model derived from noisy data.

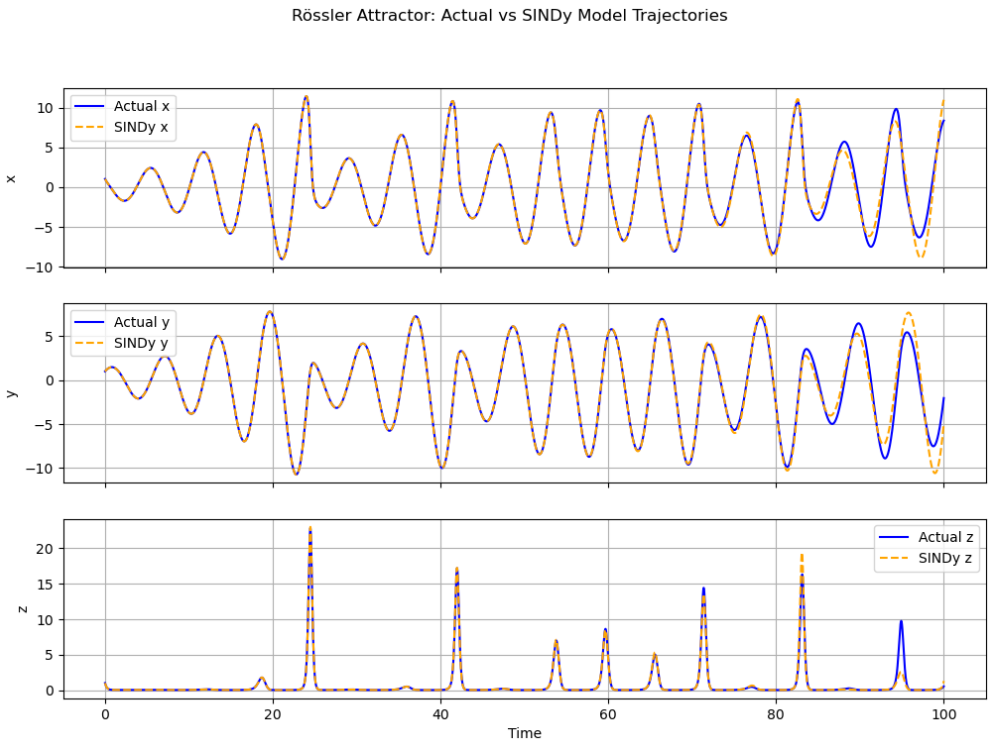


Figure 9. Rössler attractor modeling error: actuals vs. SINDy predicted.

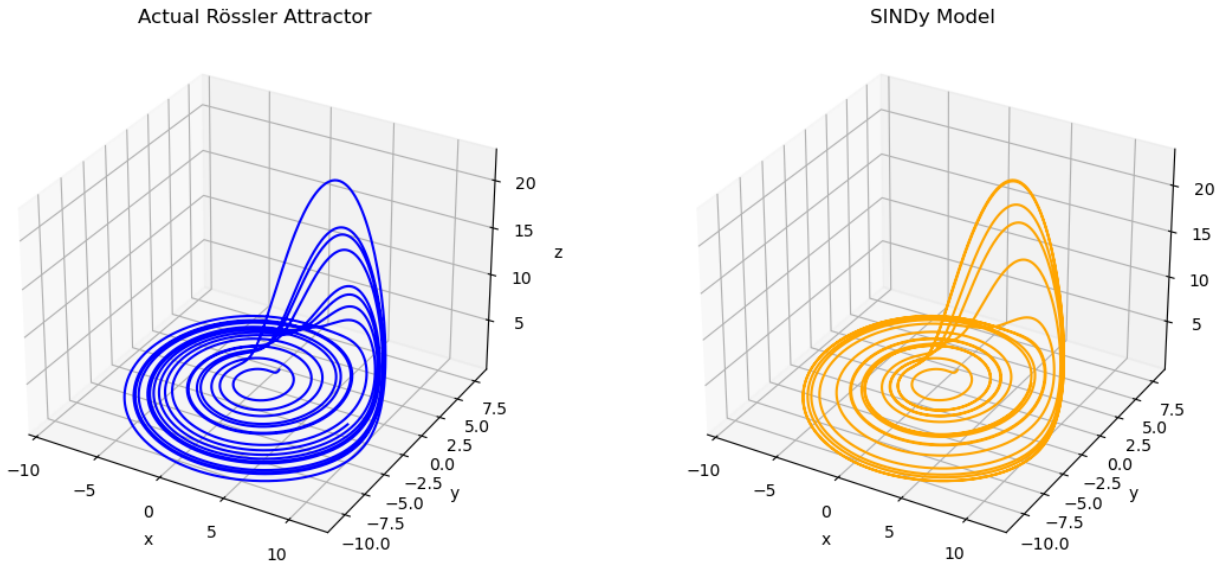


FIGURE 10. The actual solution to the Rössler attractor, and our SINDy model derived from noisy data with an increased noise profile.

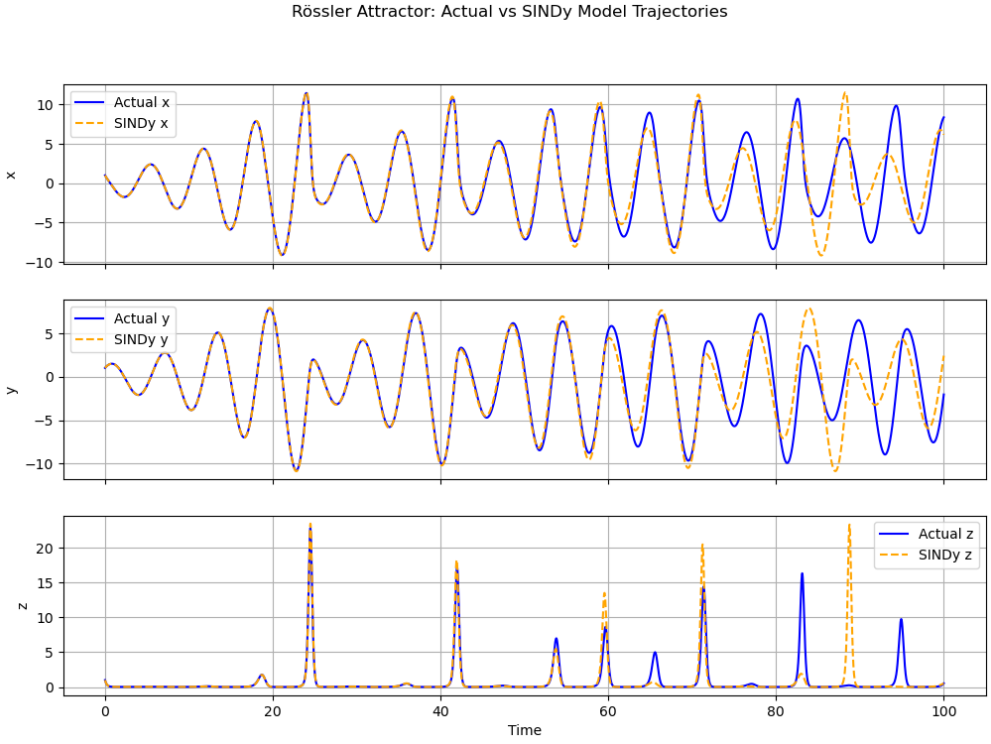


FIGURE 11. Rössler attractor modeling error with an increased noise profile: actuals vs. SINDy predicted.

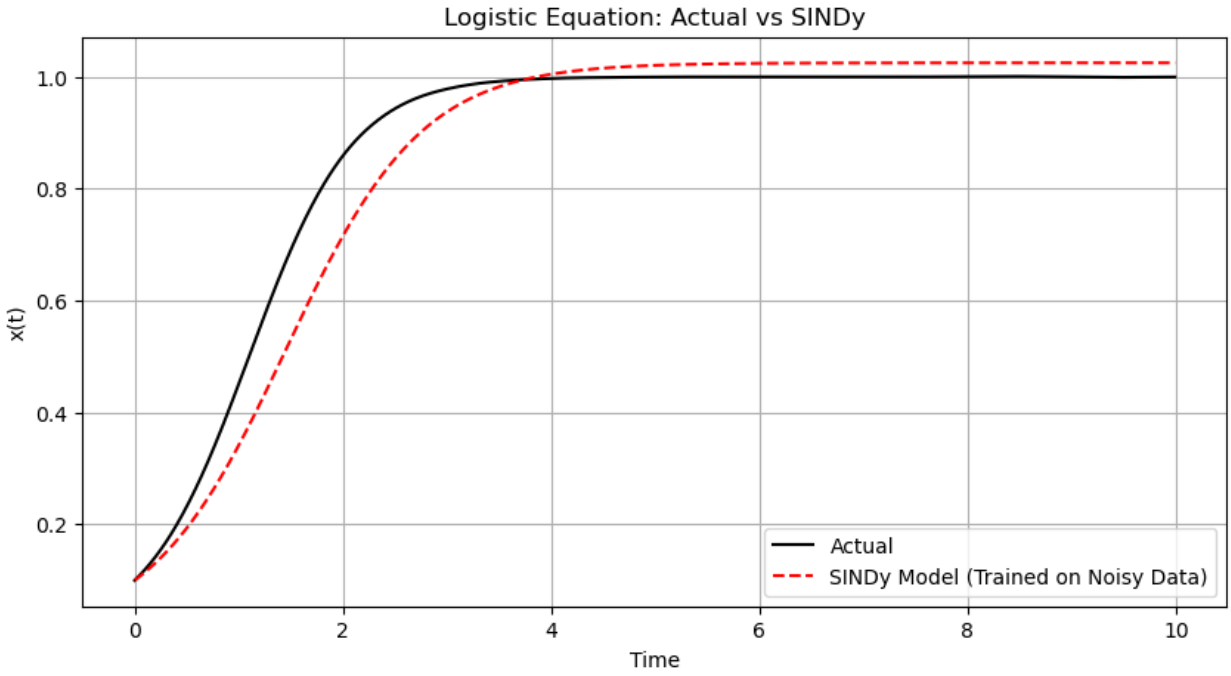


FIGURE 12. The actual solution to the logistic equation, and our SINDy model derived from noisy data.

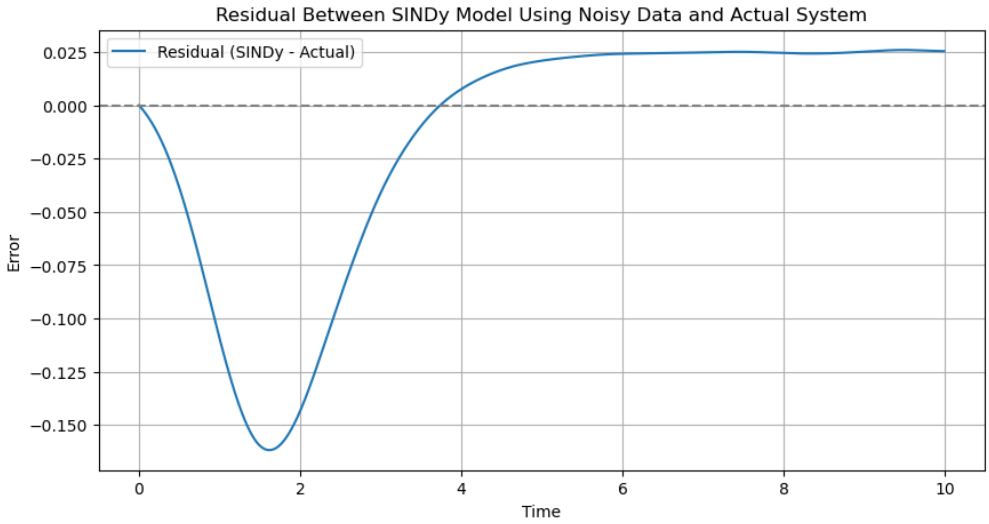


FIGURE 13. Logistic equation modeling error.

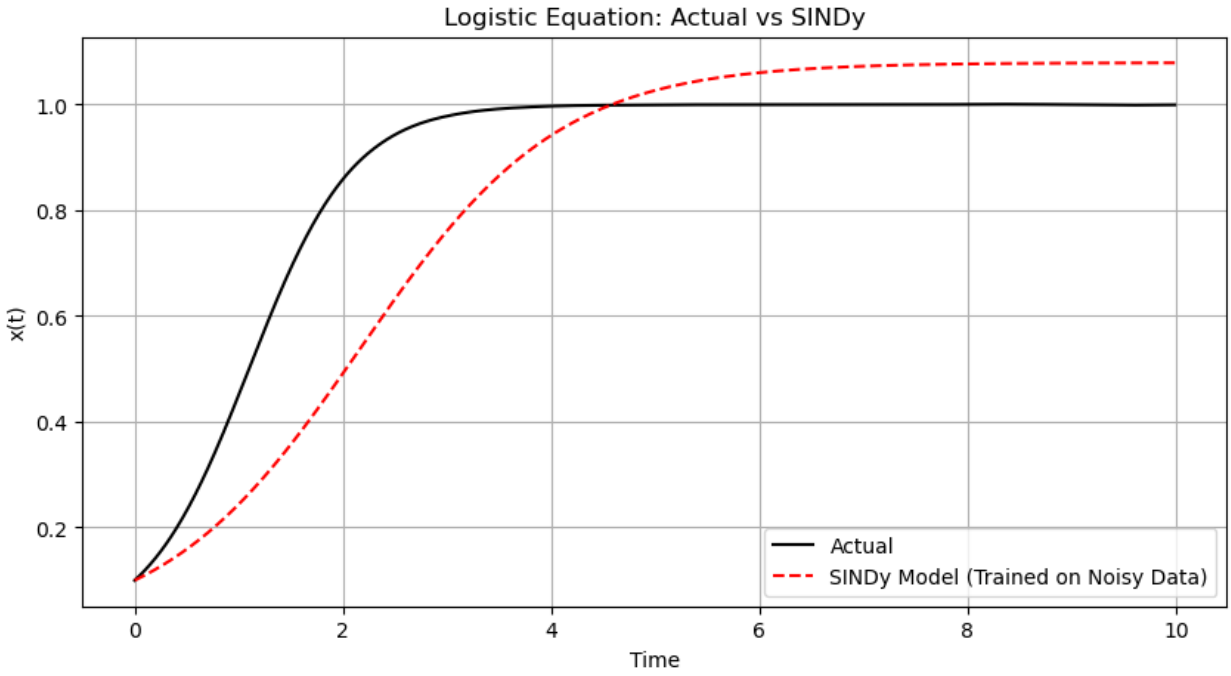


FIGURE 14. The actual solution to the logistic equation, and our SINDy model derived from noisy data with an increased noise profile.

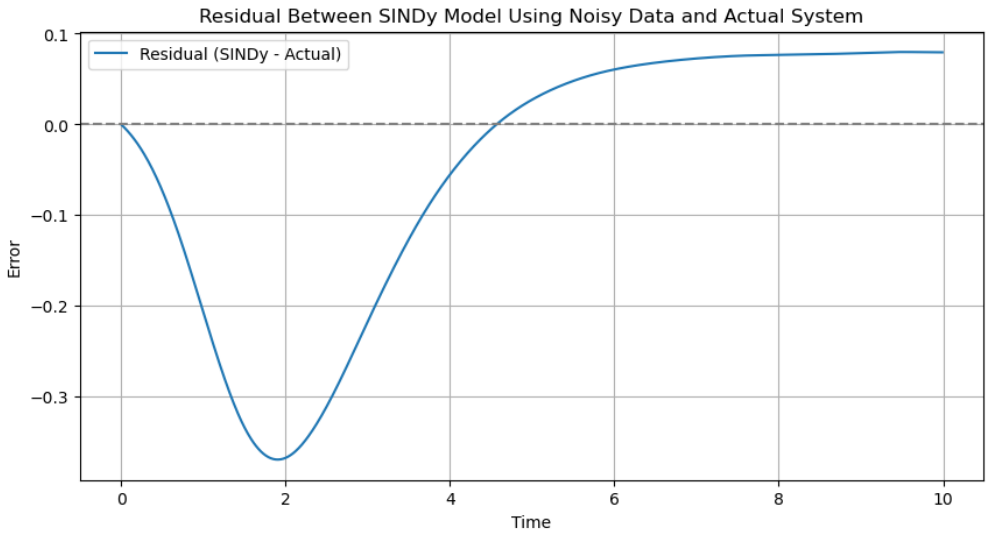


FIGURE 15. Logistic equation modeling error with an increased noise profile.

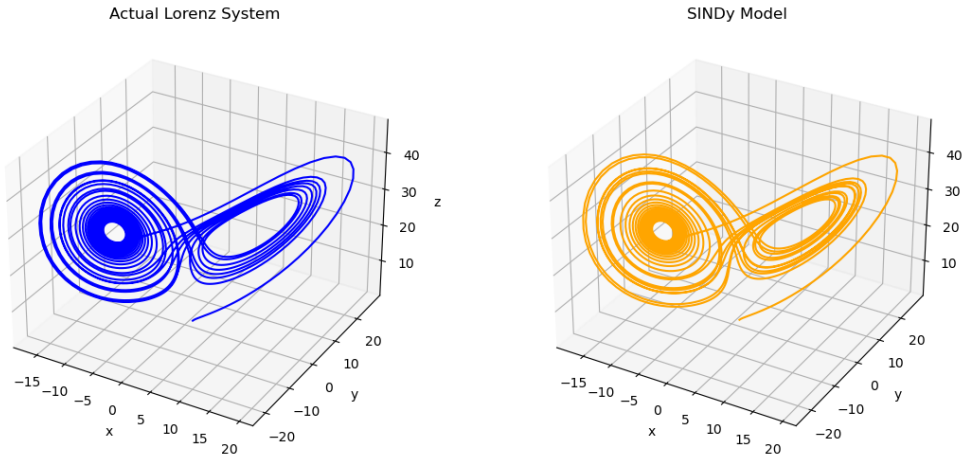


FIGURE 16. The actual solution to the Lorenz system, and our SINDy model derived from noisy data with an increased noise profile.

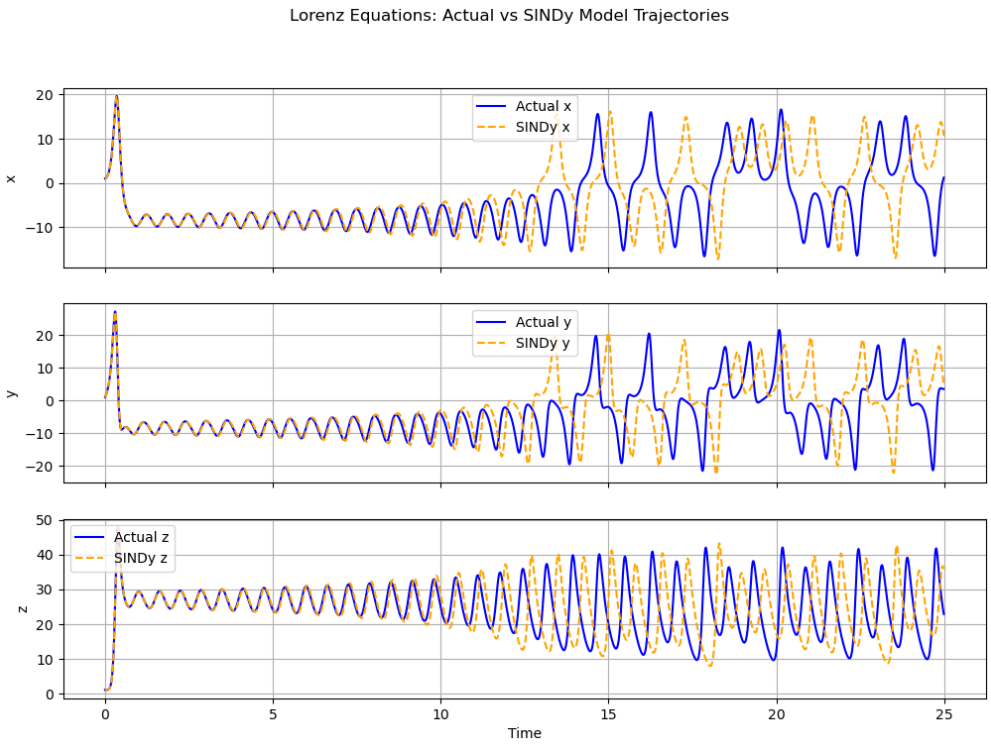


FIGURE 17. Lorenz equations modeling error with an increased noise profile: actuals vs. SINDy predicted.

APPENDIX B. CODE

The code for this project is available on GitHub. Please use the following link.

https://github.com/eklepitsch/amath575-final