Sparse Identification of Nonlinear Dynamical systems models from data using PySINDy

Sneha Manimurugan
Master of Science (Mechanical Engineering)
National University of Singapore

Acknowledgement

I would like to express my gratitude to Professor Mengaldo for his help towards this project. His advice and encouragement throughout the course of this project is much appreciated.

Abstract

This project explores the use of the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm to approach the problem of model discovery. The PySINDy package is used to apply this algorithm and discover dynamical systems models from data. This report will explain the theory behind how the SINDy algorithm works and provide analysis of the Lorenz and Rossler attractors as application examples. Towards the end, work done so far to analyse some synthetic data from a simulated seismic cycle will be described.

Introduction

Discovering dynamical systems models from data is a central challenge in mathematical physics. Dynamical systems models have been widely used to study, analyse, and predict behavior in a wide variety of application areas. These applications range from Newton's laws of classical mechanics to the Michaelis-Menten kinetics for modeling enzyme kinetics (Silva et.al,2020). There are several techniques used in the data-driven model discovery field. Some examples of these techniques include classic linear approaches, Dynamic Mode Decomposition (DMD), Koopman theory, neural networks and gaussian process regression (Kaheman et.al, 2020). Methods described above such as neural networks, are often found to face difficulties with extrapolation, are hard to interpret and do not easily apply known or partially known physics. In contrast, the SINDy method proves to generate interpretable and generalizable dynamical systems models from limited data (Champion et.al, 2020). Therefore in this project, the SINDy algorithm is used to approach the challenge of model discovery. SINDy has been commonly applied to classification of models for optical systems, fluid flows, dynamics of chemical reactions, plasma convection, structural modeling and predictive control models. The SINDy algorithm can also be expanded to recognise partial differential equations and parametric dependency of dynamical systems. It can also be used to trim corrupt data, find low-data limit control models and to integrate partially known physics and constraints. The SINDy framework depends on sparsity-promoting optimization that enables identification of parsimonious models from limited data. This results in interpretable models that avoid overfitting (Kaiser, Kutz & Brunton, 2018). Sparsity can be introduced by enabling the regression process to operate with sparsity-induced regularization (Hoffmann, Frohner & Noe, 2018). The systems analysed in this project are known to exhibit chaos within a certain range of parameters. Chaos is a generic term used to represent a chaotic dynamical system. Chaos theory, as highlighted by (Tee & Salleh, 2012), mentions that "A chaotic system is one that shows sensitivity to initial condition. Any uncertainty in the initial state of the given system, no matter how small, will lead to rapidly growing errors in predictions of the future behaviour. The system behaviour can be predicted only if the initial conditions are known to an infinite degree of accuracy, which is impossible."

Theory of SINDy

SINDy leverages the fact that many dynamical systems such as:

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

have dynamics f with only a few active terms in the space of possible right-hand side functions (Brunton S & Kutz N). For example, the Lorenz equations only have a few linear and quadratic interaction terms per equation. Lorenz is a system that exhibits chaos, which is characterized as a sensitive dependence on initial conditions. The Lorenz equations are given below:

$$\dot{x} = \sigma(y-x), \dot{y} = x(\rho - z) - y, \dot{z} = xy - \beta z \tag{2}$$

SINDy seeks to approximate f by a generalized linear model:

$$f(x) \approx \sum_{k=1}^{p} (\theta_k(x)\xi_k) = \Theta(x)\xi \tag{3}$$

with the fewest non-zero terms in ξ as possible. First, time-series data is collected from 1 and formed into a data matrix:

$$X = \begin{bmatrix} x(t_1) \\ x(t_2) \\ . \\ . \\ . \\ x(t_m) \end{bmatrix} \tag{4}$$

Then another matrix with their derivatives is formed:

$$\dot{X} = \begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ . \\ . \\ . \\ \dot{x}(t_m) \end{bmatrix} \tag{5}$$

In practice, we will aim to derive these matrices from data in X. A set of non-linear functions $\Theta(x)$ are constructed from data in X:

$$\Theta(X) = \begin{bmatrix} 1 \; X \; X^2 ... \; X^d ... \; sinX ... \end{bmatrix} \tag{6}$$

where, matrix X^d is a matrix with column vectors given by possible time-series of d-th degree polynomials in state x. The dynamical system in 1 can be represented in terms of data matrices in 5 and 6 as:

$$\dot{X} = \Theta(X)\Xi \tag{7}$$

Each column ξ_k in Ξ is a vector of coefficients determining the active dynamic terms in the k-th row in 1. A parsimonious model can then be identified to provide an accurate model fit in 7 with as minimal a number of terms as possible from Ξ. Such a model may be identified using a convex l_1 regularized sparse regression:

$$\xi_k = argmin_{\xi_k'} \left\| \dot{X}_k - \Theta(X)\xi_k' \right\|_2 + \lambda \left\| \xi_k' \right\|_1 \tag{8}$$

where, $\dot{X}_k$ is the k-th column of $\dot{X}$ and λ is a sparsity-promoting term. The sparse vectors ξ_k may be written as a dynamical system:

$$\dot{x}_k = \Theta(x)\xi_k \tag{9}$$

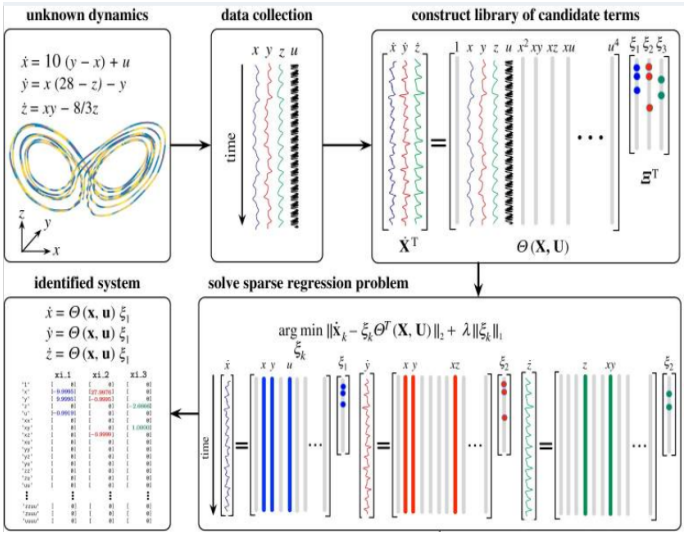

Fig. 1. Schematic of the SINDy algorithm

where x_k is the k-th element of x and $\Theta(x)$ is a row vector of symbolic functions of x, as opposed to data matrix $\Theta(X)$. Figure 1 shows the schematic how SINDy algorithm works to identify the Lorenz system. It can be seen that SINDy identifies the nonlinear Lorenz system using some measurement data. It should be noted that the algorithm also assumes that the system, as the case with many other similar systems, has relatively few terms in the governing equations. So sparsity-promoting approaches can be used to identify these system models that immediately equate sparsity with accuracy in the number of terms in model. A library of candidate nonlinear terms $\Theta(X)$ is constructed and sparse regression used to define the active terms in Θ to estimate function f.

PySINDy with Lorenz attractor

The PySINDy package was used to implement this SINDy algorithm on the Lorenz system, described by equations in 2. PySINDy is a Python package for the discovery of governing dynamical systems models from data. In figure 2 below, we can see the relationship between SINDy in 7 and the submodules of PySINDy package. The Lorenz system serves as an example of a non-linear Ordinary Differential Equation (ODE) whose solutions exhibit chaotic dynamics. The parameters used to define the Lorenz equations as in 2 are $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$. The initial condition given to generate training data is (-8,8,27). First, some training data is generated using a built-in ODE solver. The derivatives of these data are also computed by substituting the values into the Lorenz equations in 2. We then create a model(as a SINDy object) using the PySINDy package to fit these training data onto. To create this model, there are three main modeling decisions to consider:

1. The numerical differentiation method used to approximate $\dot{X}$ from X

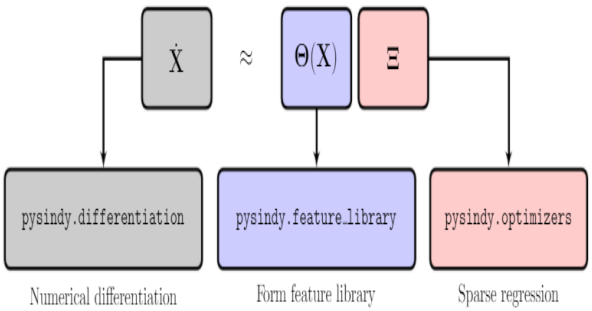

Fig. 2. Correspondence between sparse regression problem solved by SINDy (7) and sub-modules of PySINDy (Silva et.al.,2020)

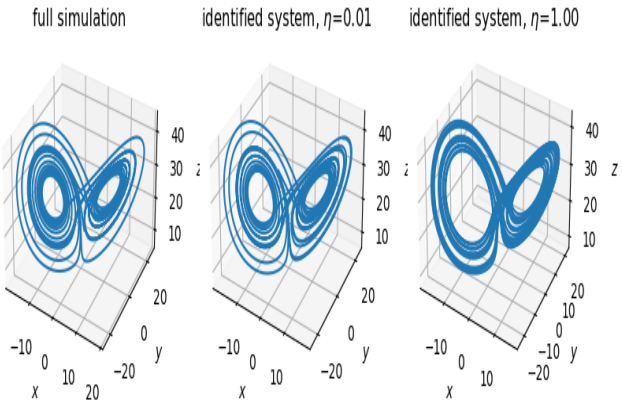

Fig. 3. Simulations of Lorenz system using training data, generated model with noise level 0.01 and noise level 1.00

2. The set of candidate functions constituting the feature library Θ
3. The sparse regression algorithm applied to solve 7 to find Ξ

Here, the default conditions for the model parameters are used initially. The conditions are:

- Numerical differentiation method (for computing $\dot{X}$ from X)- Finite difference method
- Feature library (for constructing Ξ) - Multivariate polynomials: PolynomialLibrary
- Optimizer (for performing sparse regression)- Sequentially Thresholded Least-Squares (STLSQ)

The model is then fit to the training data generated earlier using the ODE solver. In figure 3, we see 3D plots of the system simulated using training data alone and then using the SINDy model with two different gaussian noise levels introduced. As the level of noise present increases, it is noticed that the identified system becomes increasingly less similar to the original system in "full simulation".

In figure 4, 2D plots of system under influences of noise levels 0.01 and 1 are shown. It can be observed that with more noise, there is more divergence between the graphs of true data (training data) and model data.

Now that we have a trained model, we begin to assess the accuracy of this model against some test data. A new initial condition (8,7,15) is introduced and a set of test data is generated using the ODE solver again. Using this set of test data and the model, we obtain some predictions for derivatives, using function 'model.predict'. The model is also then used to compute these derivatives numerically using the finite difference method, using function 'model.differentiate'. Both sets of derivatives obtained are plotted against each other in figure 5 below. There is perfect overlap on this plot. Model scores were calculated as a measure of variance of model predicted derivatives and numerically computed derivatives. The model started off with a score of 0.976

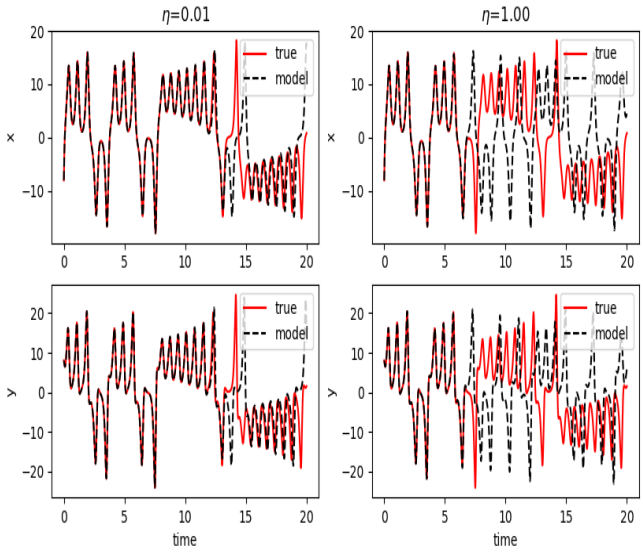

Fig. 4. x and y against time for true and model predicted data (under influence of 2 different noise levels)

accuracy but has now scored 1.000 at the point of generating plot in figure 5.

After this, we also plot the initial condition imposed for test data generation evolving with time. In figure 6 below we see plots for evolution of x,y and z with time, with and without the SINDy model.The trajectories initially match but later diverge. This can be attributed to the chaotic nature of the Lorenz system. In figure 7, we can see 3D plots of the true Lorenz system and the system identified from test

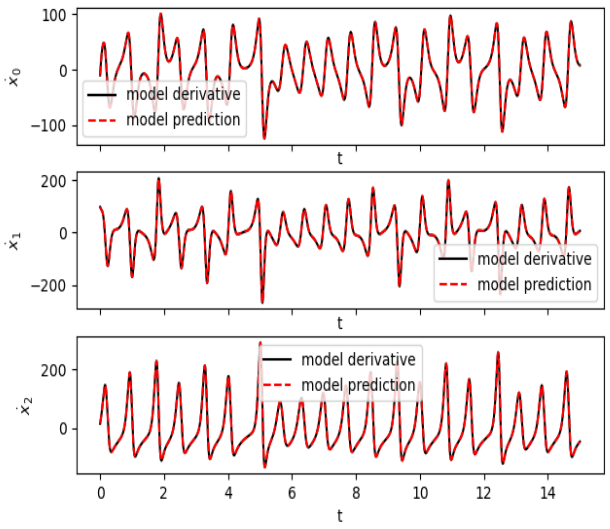

Fig. 5. Comparison of model predicted derivatives and numerically computed derivatives

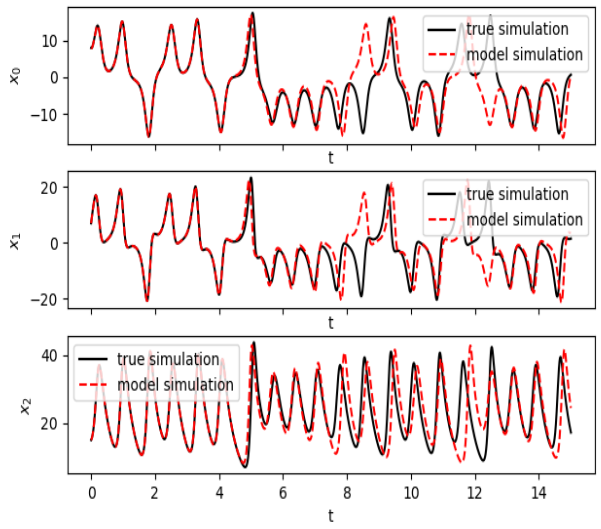

Fig. 6. Comparison of true data (test data) and model simulated data

data. The model can be considered accurate as the model generated 3D plot appears similar to that of the true system.

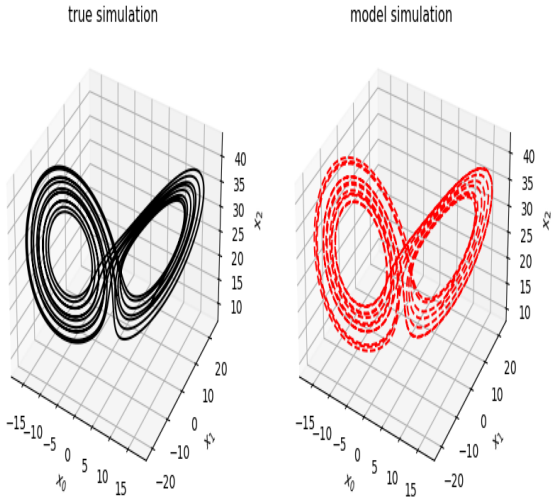

Fig. 7. 3D plots for true and model simulations of Lorenz system

PySINDy with Rossler attractor

PySINDy was again used to implement SINDy technique to analyse the Rossler system. Rossler systems were developed in the 1970s as prototype equations with minimum terms for continuous-time chaos (Gaspard, 2005). The system is inspired by geometry of flows in three dimensions and in particular by the reinjection principle. This principle is based on relaxation-type systems to present a Z-shaped slow manifold in their phase space. The Rossler equations are given below:

$$\dot{x} = -y - z, \dot{y} = x + ay, \dot{z} = bx - cz + xz \qquad (10)$$

The nonlinearity of this system phase space is minimal as there is only one quadratic term. A chaotic attractor is generated which contains one lobe, as opposed to the Lorenz attractor which has two lobes. In equation 10, we see the system equations consisting of three varibles x,y and z that evolve with time and three parameters a,b and c. The parameter values used in the simulation are a=0.2, b=0.2 and c=5.7. The initial condition given to generate training data is (3,5,0). The system is analysed the same way as the Lorenz system, using the same features to create the model. There were also some comparisons done to see how the system behaves under different types of noises with gaussian, uniform and rayleigh distributions. The model is produced with accuracy 0.999. Figure 8 shows the 3D projection plots for true Rossler system and identified systems at two different gaussian noise levels. Figure 9 shows the 3D projection plots for true Rossler system and identified systems at two different rayleigh noise levels. It can be seen that a change in type of noise distribution creates quite a difference in the model accurately identifying system close to its true representation. Figure 10 shows how true and model values vary over the course of time for x and y. When compared to the

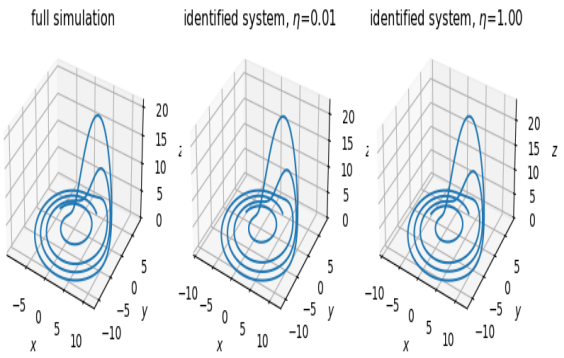

Fig. 8. 3D plots for true and model simulations of Rossler system for two different noise levels (gaussian)

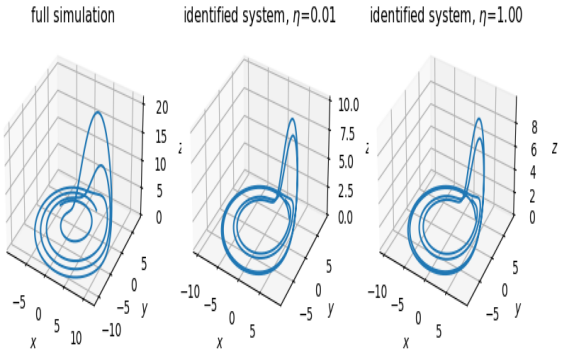

Fig. 9. 3D plots for true and model simulations of Rossler system for two different noise levels (rayleigh)

Lorenz system model, this model seems to be more accurate in matching the true values of x and y as they evolve with time. This accuracy is maintained and seems invariant to increase in noise level introduced. We can see in figure 11 that the numerically computed derivative almost perfectly match the model computed derivatives. Figure 12 shows the evolution of test data under a new intial condition imposed, (2,-5,10). Figure 13 shows matching results of the identified system to the true system under use of test data. All the results presented for both the Lorenz and the Rossler systems prove well that the PySINDy tool accurately identifies nonlinear dynamical system models from a set of generated data.

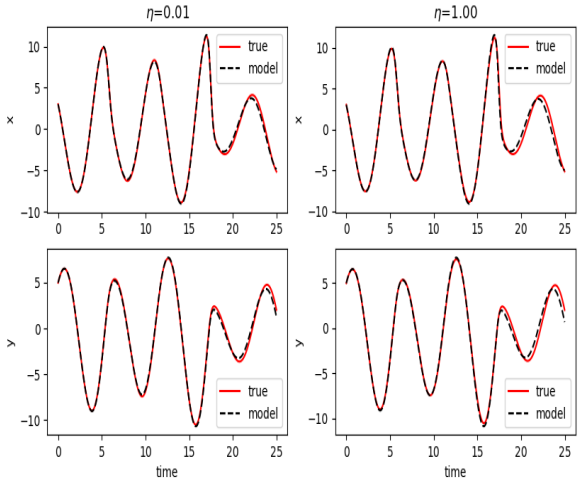

Fig. 10. x,y against time for two different noise levels (gaussian)

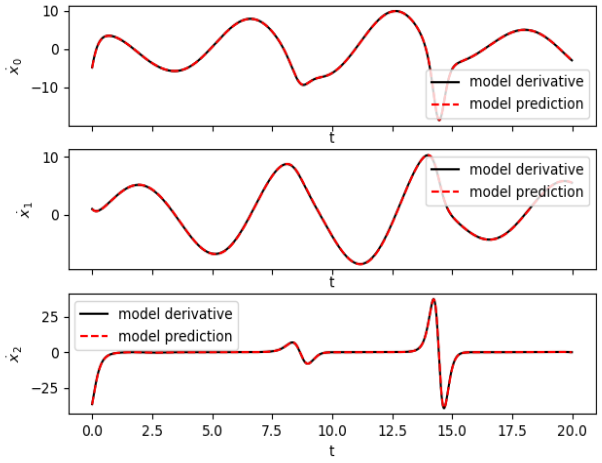

Fig. 11. Comparison of derivatives computed and generated

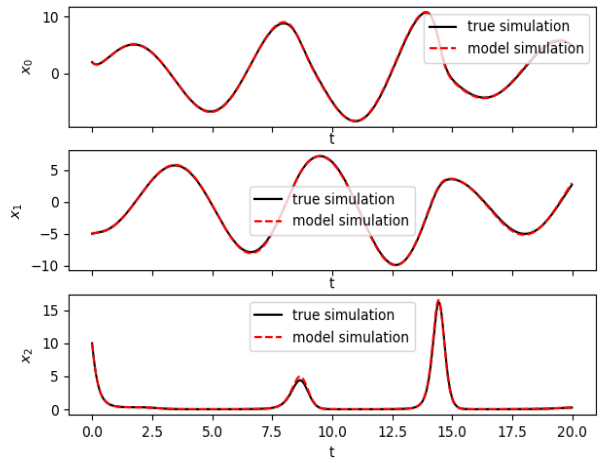

Fig. 12. 2D plots of x,y against time using new initial condition for test

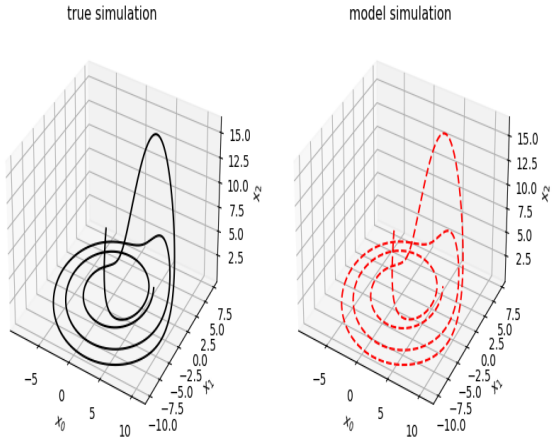

Fig. 13. 3D plots of Rossler system under new test initial condition and true model

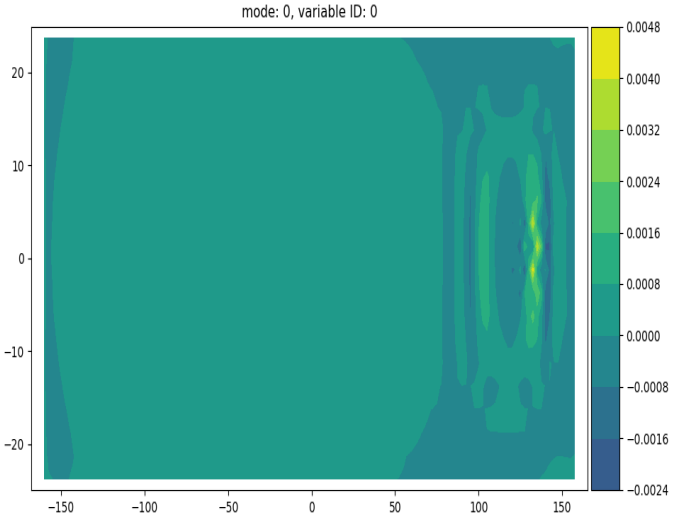

Fig. 14. First mode

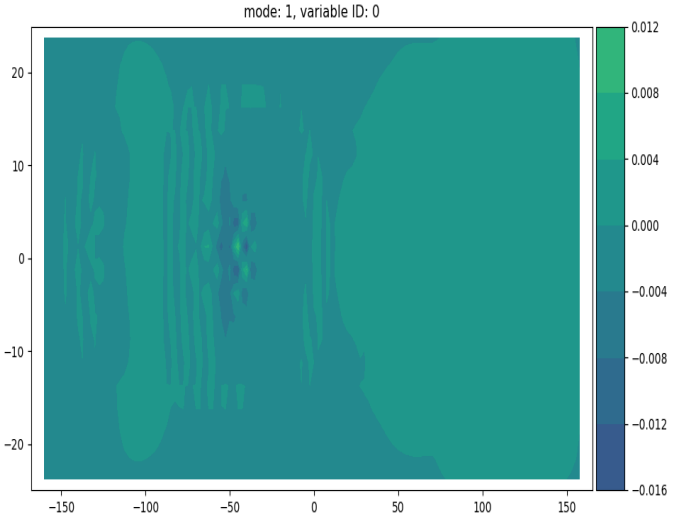

Fig. 15. Second mode

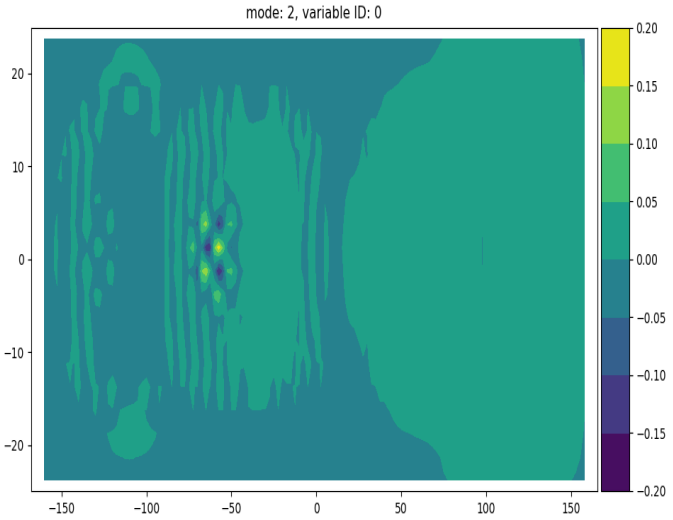

Fig. 16. third mode

Experimenting with earthquake data

A set of synthetic data from a simulated seismic cycle was experimented with in this project. The dataset has been generated with the notion of mimicking the Cascadia subduction region and it contains slip potency data from a numerical simulation. From previous studies, this data could reproduce some important features like the segmentation that we observe in nature. The model used for these simulations is highly non-linear and it involves several interacting patches. Despite these complications, from performing a first dimensional analysis, the system could be described by a reduced number of degrees of freedom. A question then arose if we could deduce an approximation of the original equations in an augmented space. It was intended to use PySINDY to analyse the data for the above said purpose. During the implementation of PySINDy, there were some difficulties faced with regards to the sparsity parameter being too big and dealing with errors in proper sizing of the training data set for use when creating the PySINDy model. To mitigate the sizing problem, we decided to use a Spectral Proper Orthogonal Decomposition (SPOD) library written by Professor Mengaldo to extract a set of 3 spatial modes that oscillate in time at a single frequency. The idea was to then pass in these three modes only as training data to create the PySINDy model. From SPOD, results are obtained as a set of eigenvalues per each frequency computed and a set of modes per each frequency computed. Figures 14, 15 and 16 show the three modes extracted from one tranche of given earthquake data. Default parameter settings were used, except the length of Fast Fourier Transform (FFT) blocks used was 64 instead of 32 to speed up analysis time. Figure 17 shows a 3D plot for training data (true system) that was seen by the model PySINDY. This could be due to the SPOD returning some modes that are actually not physical due to the non-uniform time stepping.

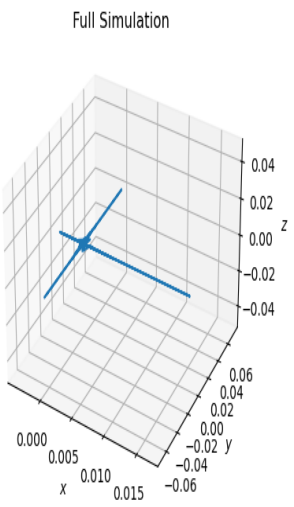

Fig. 17. Plot of true system for PySINDy

Conclusion and future work

This project used PySINDY to successfully perform nonlinear dynamical system identification on the Lorenz and Rossler attractors. The results obtained also prove mostly that the SINDy algorithm has managed to identify systems as close to the true system models as possible. The earthquake data simulates a real-world dataset that illustrates an ensemble of points with specific relationships between them. Therefore through this project, we could see applications of numerical methods taught in class to solve ordinary differential equations. Due to time constraints, the report has been written with findings thus far but there still is work to be done on analysing the earthquake data with PySINDy. This will then allow us to truly understand and test the ability of the SINDy algorithm on real-world data.

References

[1] Silva B, Champion K Quade M, Loiseau J, Kutz N & Brunton S *PySINDy: A Python package for the Sparse Identification of Nonlinear Dynamics from Data*, 2020.

[2] Kaheman K, Kutz N & Brunton S, *SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics*, 2020, The Royal Society Publishing.

[3] Champion K, Zheng P, Aravkin A, Brunton S & Kutz N, *A Unified Sparse Optimization Framework to Learn Parsimonious Physics-Informed Models From Data*, 2020, IEEE.

[4] Hoffmann M, Frohner C & Noe F, *Reactive SINDy: Discovering governing reactions from concentration data*, 2019, The Journal of Chemical Physics.

[5] Tee L & Salleh Z, *Dynamical Analysis of a Modified Lorenz System*. 2012, Journal of Mathematics.

[6] Kaiser E, Kutz N & Brunton S, *Sparse identification of nonlinear dynamics for model predictive control in the low-data limit*, 2018, The Royal Society Publishing.

[7] Gaspard P, *Rossler systems*. 2005, Encyclopedia of Nonlinear Science.

[8] PySINDy documentation, *An introduction to Sparse Identification of Nonlinear Dynamical systems (SINDy)*.

[9] PySPOD package by Professor Mengaldo, "https://github.com/mengaldo/PySPOD"

[10] Brunton S & Kutz N, *Data-Driven Science And Engineering: Machine Learning, Dynamical Systems, And Control*.