4-2019

Training Set Density Estimation for Trajectory Predictions Using Artificial Neural Networks

Zachary Reinke
Western Michigan University

TRAINING SET DENSITY ESTIMATION FOR TRAJECTORY PREDICTIONS USING ARTIFICIAL NEURAL NETWORKS

by

Zachary Reinke

A thesis submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
Mechanical Engineering
Western Michigan University
April 2019

Thesis Committee:

Jennifer Hudson, Ph.D., Chair
Richard Meyer, Ph.D.
Damon Miller, Ph.D.

ACKNOWLEDGMENTS

First and foremost I want to thank my Savior Jesus Christ for giving me the opportunity to pursue and achieve my dream of earning a degree in engineering. I would like to thank my wonderful wife Karla for all the love and support (moral and financial ◡̈). You've been amazing! I would like to thank my Mother for giving me a passion for learning and exploring the world around me and for always supporting my dreams. I would like to thank Dr. Hudson for taking me on as a graduate student. Your guidance and patience have made this experience exceptional and has taught me invaluable lessons! I would like to thank Dr. Miller for believing in me and giving me so many opportunities to learn and grow during my time at WMU. I would like to thank Dr. Meyer for the many hours spent answering my crazy qestions about math, engineering, and Legos. And lastly, I would like to thank the faculty of the MAE department at WMU. I'm so grateful for all the time and effort that you have invested into my education.

Zachary Reinke

TRAINING SET DENSITY ESTIMATION FOR TRAJECTORY PREDICTIONS USING ARTIFICIAL NEURAL NETWORKS

Zachary Reinke, M.S.E.

Western Michigan University, 2019

Demand on earth orbiting object surveillance systems is increasing as more equipment is put into orbit. These systems rely on predictive techniques to periodically track objects. The demand on these systems may be reduced if object trajectories could be predicted further into the future. This research developed techniques that analyze state trajectory data to develop scalable training sets used for training artificial neural networks (ANNs) to predict trajectories of a dynamic system. These methods use multi-variable statistics to analyze data energy content to provide the ANN with low density, feature-rich, training data. The developed techniques have been shown to increase ANN prediction accuracy while reducing the size of the training set when applied to a linear dynamic system. These methods may find future application in predicting satellite trajectories.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There is an increasing number of objects orbiting the earth. These objects vary from active satellites to space debris. It has become an increasing challenge to track these objects. The Space Surveillance Network (SSN), part of the United States Space Command (USSPACECOM), tracks more than 8,000 objects via ground based sensors [1]. These objects range from large satellites to small spent rocket parts. The SSN currently tracks objects that are 10 centimeters or larger. Approximately 7% of these objects are operational satellites and the rest are debris [2]. Space debris is becoming a growing concern for the hazard it poses for current and future space missions. Every year more debris is introduced into low earth orbit, increasing the risk of collision with active space missions. At high relative velocities, even collisions with small objects can result in significant damage [3].

Due to the large number of space objects the SSN cannot track objects continuously. Instead, the SSN uses a predictive technique to monitor space objects. It spot-checks the objects periodically to identify them and record their state dynamics. This spot-check technique is used since the number of sensors, geographic distribution, capability, and availability of the SSN sensors limits the frequency at which objects can be tracked. Once an object is tracked, the network uses a predictive technique to periodically identify the object based on the system dynamic previously measured. The accuracy of this identification relies heavily on the prediction method and also on the accuracy of the SSN sensors.

A constellation of 48 small satellites in low earth orbit (LEO), known as Flock2K, was used as a case study [4]. Two-line elements (TLE) sets form a standard convention for organizing satellite tracking data [5]. Gathered from the Combined Space Operations Center (CSpOC) catalog, TLEs were used to estimate the update frequency for the Flock2K tracking information [6]. The elapsed time between measurements was found to be several hours. From this analysis it was

also observed that the satellites may not have always been accurately identified. Sharp jumps in the data indicate that the prediction model may have misidentified satellites.

Several research projects have sought to improve tracking accuracy using statistical and model driven approaches [7, 8, 9]. Methods for state estimation of orbiting objects for improved tracking have also researched [10]. In recent years artificial neural networks (ANN) have been applied to many different types of problems. Advances in computers have made their application widespread with great success [11, 12]. ANNs have also made an emergence in aerospace applications, such as adaptive allocation of sensing resources for improved tracking and space situational awareness [13]. ANNs could be applied to trajectory predictions for improved tracking accuracy. An ANN may be trained to model the perturbations acting on the system dynamics of an orbiting object. This could provide a way to more accurately perform state trajectory predictions for passive objects orbiting the earth. Figure 1.1 shows the orbital state elements of a satellite in orbit. This could be a good alternative to conventional tracking methods and provide better prediction models than traditional techniques .

Simple orbital trajectories can be modelled using the equations that describe two masses in mutual orbit around one another [14]. While this forms the foundational physics that govern two orbiting bodies, this is an incomplete model of the orbital trajectory. Additional factors such as atmospheric density, solar pressure, and the magnetic field of the earth influence the orbits. These factors, referred to as perturbations, form complex motions caused by forces other than the gravitational force [15]. One of the biggest challenges of orbital trajectory predictions comes from predicting the perturbations acting on the satellite. There has been research conducted to use ANNs to model the behavior of the individual constitutes of these perturbations such as solar activity, electron flux, atmospheric density, and geomagnetic activity, to name a few [16, 17, 18, 19].

A possible solution may be to implement an ANN designed to add unmodeled perturbations to a baseline trajectory prediction [20]. The input to the ANN is a simulated trajectory based on known system dynamics and perturbations that are well modelled. From this incomplete simulated trajectory, the ANN will add missing high level perturbations. This is achieved by training the ANN output on real satellite trajectory data.

Training data is a vital element in developing an ANN. The quality of the training data sets a limit for the performance the ANN can achieve. Training data must include sufficient information

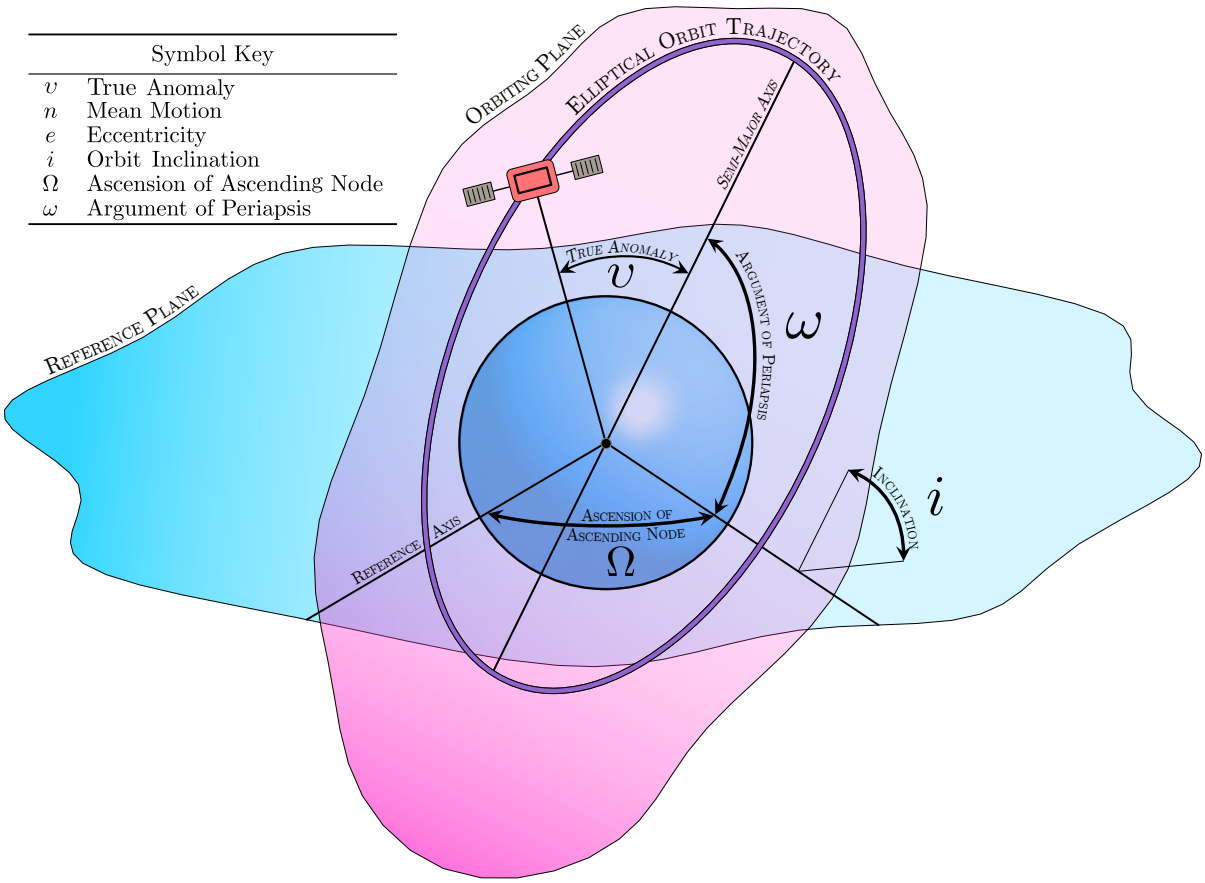


Figure 1.1: Orbital State Elements

about the system that is to be emulated. The system must be well exercised to properly extract features of its behaviour. Assembling training data that is dense with information about system behaviour will aid in increasing the accuracy of the resultant ANN. The goal of this research is to develop methods to assemble effective and efficient training data for training an ANN for trajectory predictions. These methods will use data driven analysis to analyze the information content of real-world trajectory data. From this analysis, the training set density can be reduced while still maintaining prediction accuracy. Training an ANN using low density and feature rich data could allow the network to train faster without loss of performance. This could reduce the overall time to train and update the ANN. The time saving could be significant as complex systems often require a large amount of information to train.

Several different architectures may be considered for trajectory predictions. Recurrent Neural Networks (RNN) are often used for modelling the behavior of dynamic systems [21]; however,

this application does not call for modelling the dynamic system but rather to add information to the system trajectory. Another common neural network architecture is the Multi-Layer Feed-Forward (MLFF) neural network [22]. This architecture is often used for non-linear regression and its application is straight forward. Because of its straight forward application, a MLFF will be used for this research. The goal here it to test the proposed methods and lay a baseline performance result.

CHAPTER 2

MODELING PERTURBATIONS WITH AN ANN

The state trajectory of an orbiting system can be described in varying complexity. The simplest model for an orbiting object is described by the "Two Body Problem" [14]. Equation (2.1) describes the motion of one orbiting body relative to another

$$\mathbf{F} = \frac{G m_1 m_2}{r^2} \hat{\mathbf{r}} \tag{2.1}$$

where $\mathbf{F}$ is the force acting on the two bodies, $\hat{\mathbf{r}}$ is a vector pointing from one center of mass to the other, r is the distance between the two mass centers, G is the gravitational constant, and m_1 and m_2 are the mass of each body. The physics described by Equation (2.1) are the basis of every model for two body motion. However, there are perturbations present in a real-world system that Equation (2.1) does not capture and thus much of the information about the system is missing. This type of model shall be referred to as a *low fidelity* model. The *high fidelity* model will be comprised of real-world data which includes all the perturbations present in that data.

In this work, an ANN is used to provide a high fidelity state estimate $\vec{X}_H$ given a low fidelity state estimate $\vec{X}_L$, its derivative $\dot{\vec{X}}_L$, and the time of the estimate t. The ANN weights are adapted to minimize the error E between the its output $\vec{O}$ and p examples of $\vec{X}_H$ from observed data, where

$$E = \frac{1}{P} \sum_{p} \left\| \vec{O}\left(\vec{X}_{Lp}, \dot{\vec{X}}_{Lp}, t, \mathbf{W}\right) - \vec{X}_{Hp} \right\|^2. \tag{2.2}$$

The state $\vec{X}_{Lp}$ and $\dot{\vec{X}}_{Lp}$ are computed using a low fidelity model. a successfully trained ANN can be used to provide a high fidelity estimate from a low fidelity estimate based on relationships captured in the p examples of the training data. In essence, the ANN is 'learning' the error $\vec{e} = \vec{X}_H - \vec{X}_L$. The trained ANN can be repeatedly applied to estimate state trajectories.

Figure 2.1 shows the state space graph of the dynamic system depicting the input and output trajectories. A similar approach was used by Peng and Bai [20] for testing the orbital prediction accuracy limits of machine learning. More accurate low fidelity information reduces the difference between the high and low fidelity estimates, and therefore the fewer features the ANN is required to learn.

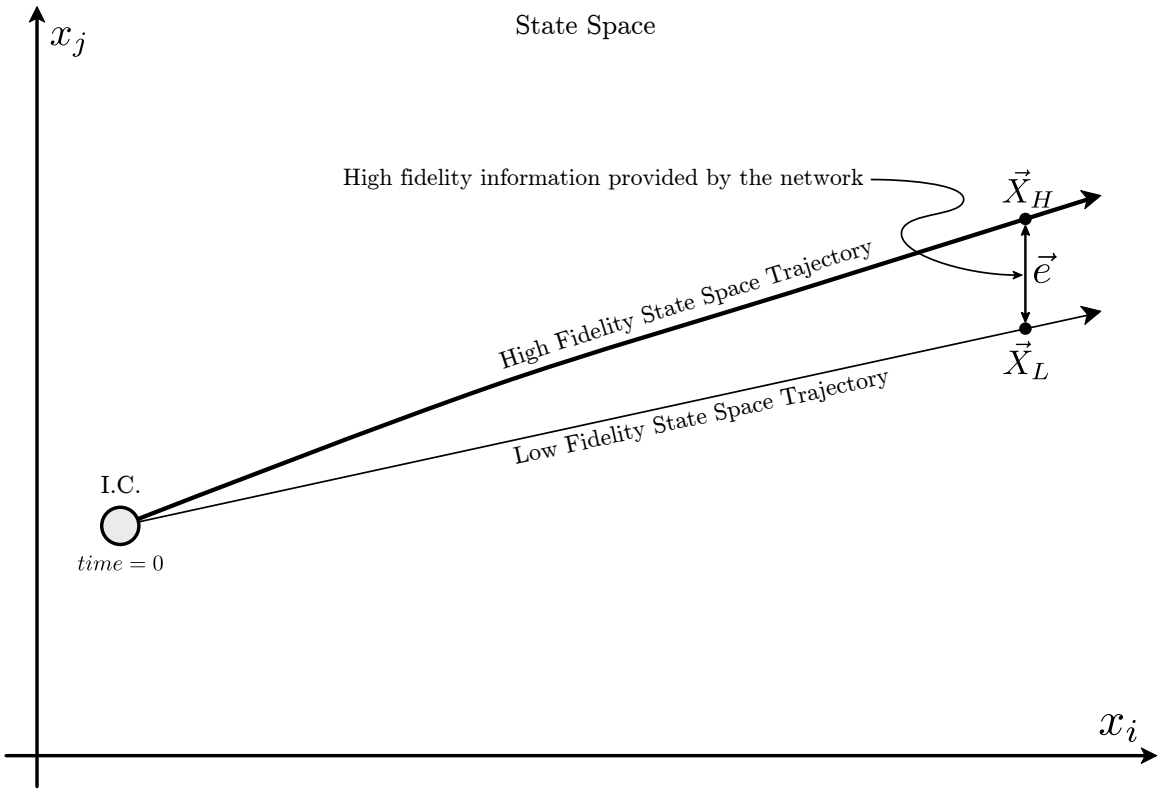


Figure 2.1: High and Low Fidelity State Estimates

CHAPTER 3

TRAINING DATA

Obtaining in-world data for training an ANN can be a challenge. There is a limit, not only the amount of data, but also on the quality of the data. It can be a far greater challenge to acquire and condition the data needed for training than it is to design and train the ANN. In these cases, it may be beneficial to reduce the size of the training set. This may be required if there is limited data or if the computational expense is high for processing the data. It may also be beneficial to reduce the training set in order to speed up the training of the network.

Each trajectory is initiated by an initial condition. The system dynamics are propagated forward in time tracing out the state trajectory that originates from a particular set of initial conditions (ICs). The neural network should be able to predict high fidelity states along the complete set of state trajectories that encompass the system dynamics initiated from any set of valid ICs within a state's valid range of values. The question then becomes, what size should the step size between data points be for an effective training set? There are two different step sizes to consider. The first is the time step taken when propagating a state trajectory forward in time. This step size must be small enough to preserve system behavior. The second is the step size between the ICs that initiate the state trajectories. It too must be small enough that the system behavior is captured. It must also be small enough that the ANN can accurately interpolate the system behavior when providing information that falls between the points that were used in training. Figure 3.1 shows a local grid of initial conditions in 2 dimensional state space. While depicted here in 2 dimensions, this discussion extends to n dimensional state space. A state trajectory is propagated from each IC giving the content of the training set. The two step sizes are shown. One is the time steps of the trajectory, Δt, and the other is the distances between the IC grid, Δ_i and Δ_j. The ANN will be trained with trajectory data initiated by ICs 1 through 4. By increasing Δ_i and Δ_j the training set density is reduced. However, increasing these step sizes will directly affect the network ability to

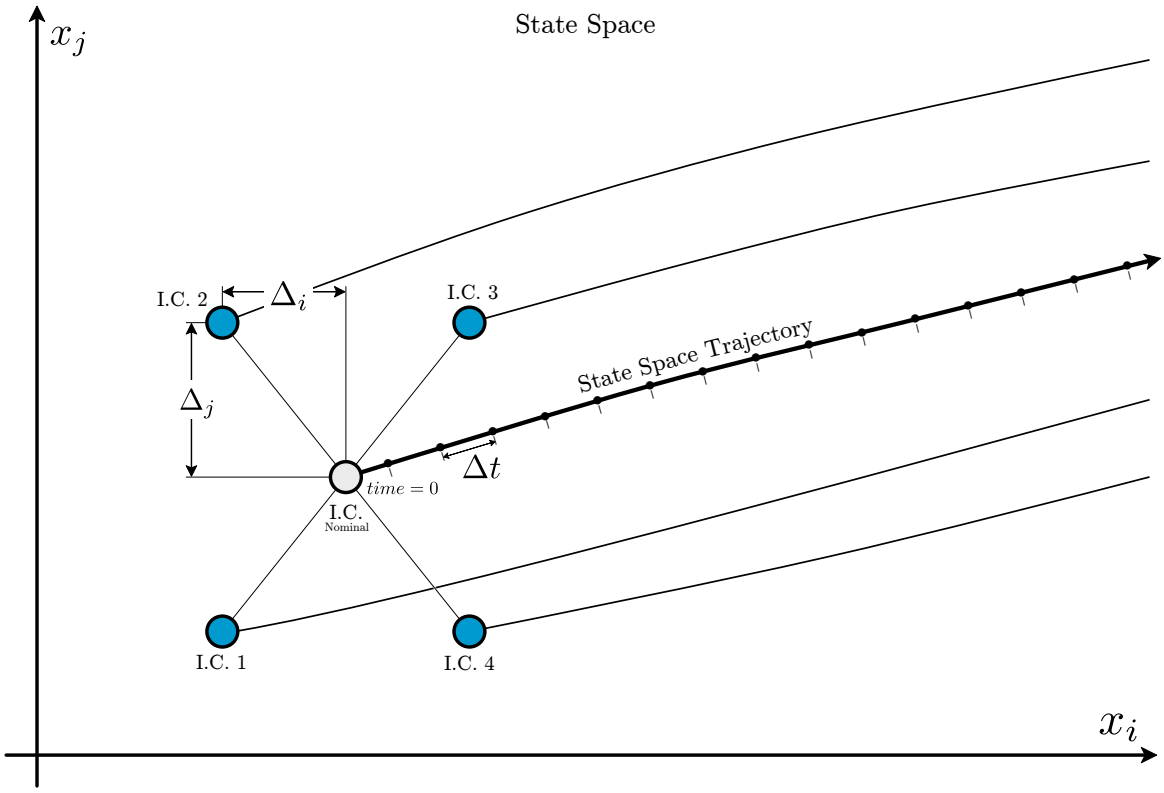


Figure 3.1: State Space Trajectories

reproduce the trajectory initiated by the nominal IC. The nominal IC represents a ”worst-case” in terms of trajectory prediction as it falls at a point farthest from all ICs used for training. Therefore, trajectory data initiated from the nominal IC will be one metric used to measure the performance loss of lowering the training set density. In this work the time step size is fixed while methods to estimate an effective spatial step size are investigated.

CHAPTER 4

SCALING THE TRAINING DATA DENSITY

One method for setting the step size for the grid of initial conditions would be to consider the variance of each state. It is assumed that a state with a higher variance would have more information about the system characteristics than one with a lower variance. Therefore a higher variance would suggest that smaller spatial steps should be taken for that state. Taking smaller steps would provide the training set with more information about the system dynamics and therefore present the network with a richer set of data when training. It is also assumed that a state with a lower variance would suggest that larger spatial steps for that state could be taken. This could be very useful since it would provide some measure of the minimum size for the training set that still encompasses the system characteristics.

One problem with using the variance of an individual state is that there is no consideration for the interaction between the states of the system. Each state would be analyzed in isolation. To understand the multivariate variance of the system, the covariance matrix will be analyzed using *Singular Value Decomposition* (SVD). The SVD is a matrix decomposition that provides a hierarchical representation of the data in terms of an alternate coordinate system defined by the dominant correlations within the data.

4.1 Example: Multivariate Variance using SVD

Consider the state matrix $\mathbf{X}$ (4.1). The columns of $\mathbf{X}$ contain time samples of the states. Figure 4.1a shows a scatter plot of the example data.

$$\mathbf{X} = \begin{bmatrix} | & | \\ \mathbf{x}_i & \mathbf{x}_j \\ | & | \end{bmatrix} \tag{4.1}$$

Once the state matrix is assembled the column-wise means are calculated. Here n is the number of elements in $\mathbf{x}_i$ and $\mathbf{x}_j$. Thus

$$\bar{x}_j = \frac{1}{n}\sum_{l=1}^{n} x_{lj} \tag{4.2}$$

$$\bar{x}_i = \frac{1}{n}\sum_{l=1}^{n} x_{li} \tag{4.3}$$

$$\bar{\mathbf{X}} = \begin{bmatrix} | & | \\ 1 & 1 \\ | & | \end{bmatrix} \odot \begin{bmatrix} \bar{x}_i & \bar{x}_j \end{bmatrix}. \tag{4.4}$$

Subtracting the column-wise mean from the state output matrix $\mathbf{X} - \bar{\mathbf{X}}$ yields

$$\mathbf{B} = \mathbf{X} - \bar{\mathbf{X}} = \begin{bmatrix} | & | \\ \mathbf{b}_i & \mathbf{b}_j \\ | & | \end{bmatrix} \tag{4.5}$$

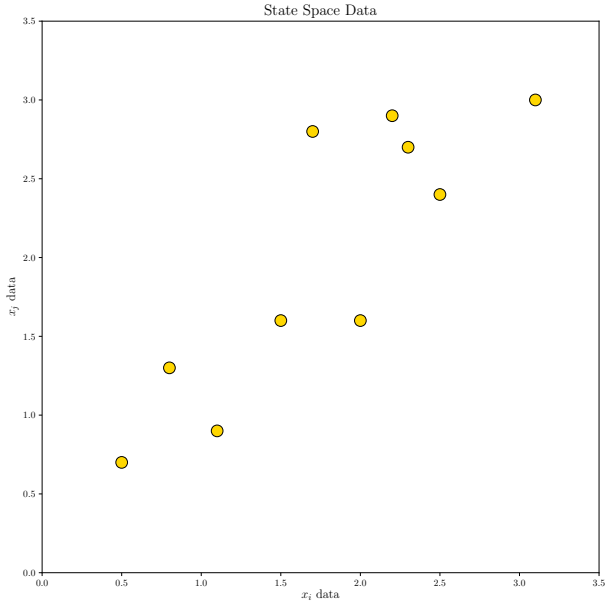
and centers the data about the mean as shown in Figure 4.1b. The covariance between two variables is calculated as

$$Covariance \;=\; \frac{1}{n-1}\sum_{l=1}^{n}\left(x_{li} - \bar{x}_i\right)\left(x_{lj} - \bar{x}_j\right). \tag{4.6}$$

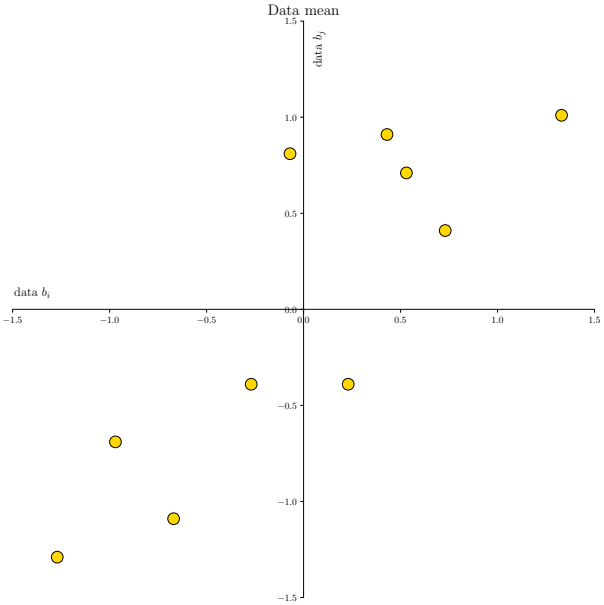
The covariance matrix of X is

$$\mathbf{C} \;=\; \frac{1}{n-1}\begin{bmatrix} - & \mathbf{b}_i & - \\ - & \mathbf{b}_j & - \end{bmatrix}\begin{bmatrix} | & | \\ \mathbf{b}_i & \mathbf{b}_j \\ | & | \end{bmatrix} \;=\; \frac{1}{n-1}\mathbf{B}^T\mathbf{B}. \tag{4.7}$$

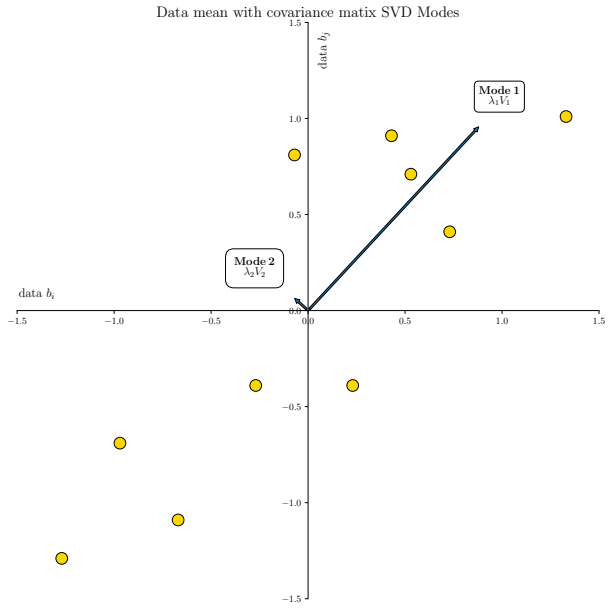
λ is a diagonal matrix whose values are the eigenvalues of $\mathbf{C}$, (Table 4.1). $\mathbf{V}$ is a matrix whose columns are the eigenvectors of $\mathbf{C}$, (Table 4.2). Note that taking the eigenvalues and eigenvectors of the covariance matrix is part of calculating the SVD of the covariance matrix.



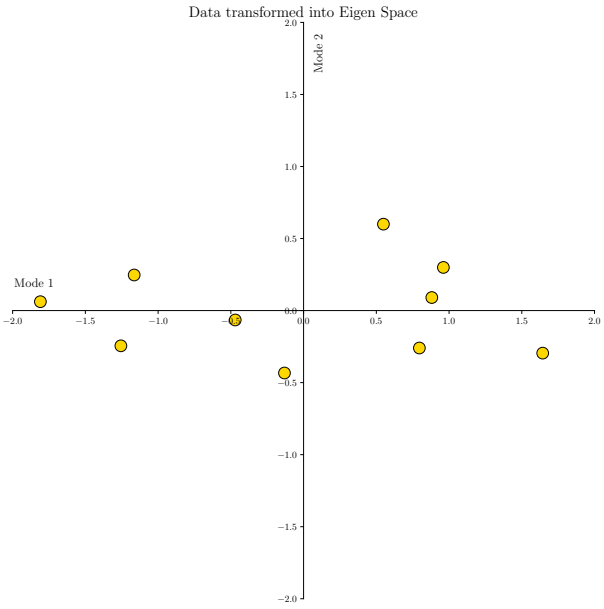
(a) State Space Output (Columns of X)



(b) Columns of B



(c) Modes of the covariance matrix of X



(d) State variable outputs in Eigenspace

Figure 4.1: SVD Example Data

Table 4.1: Eigenvalues

λ_1	1.309
λ_2	0.103

Table 4.2: Eigenvectors

V_1	$[0.676, 0.736]$
V_2	[-0.736, 0.676]

The eigenvectors of the covariance matrix are an orthogonal set of basis vectors. They have also been normalized as unit vectors. This makes the matrix **V** a rotation matrix. This rotation is a transformation from the original coordinate system. The corresponding eigenvalues are the variances of the data projected onto these eigenvectors. Figure 4.1c shows the centered state space data with the scaled eigenvectors super-imposed onto the plot. Notice how the vector formed by $\lambda_1 V_1$ appears to lie along a line going through the most densely populated part of the plot. Also notice how the vector formed by $\lambda_2 V_2$ is orthogonal and much shorter than the other vector.

As **V** is a rotation matrix, the state data can now be transformed into the eigenspace using

$$\mathbf{B}_{Rot} = \mathbf{BV}. \tag{4.8}$$

The columns of $\mathbf{B}_{Rot}$ are the state variable outputs transformed into eigenspace Mode 1 and Mode 2. Figure 4.1d shows a plot of the state variable outputs in eigenspace. If we now take the variance of the columns of $\mathbf{B}_{Rot}$ we find that they are the eigenvalues of the covariance matrix of **X** as seen in Table 4.3. The eigenvalues and eigenvectors of the covariance matrix provide a way to calculate

Table 4.3: The Column Variance of $\mathbf{B}_{Rot}$

$variance(Mode1)$	1.309
$variance(Mode2)$	0.103

the variance of combinations of states rather than considering each state in isolation. This may provide a more meaningful way to calculate the training set step size.

CHAPTER 5

DYNAMIC SYSTEM

The dynamic system for an orbiting body is harmonic in nature. An accurate model for this system is complex and non-linear but might be approximated with a linear model. This might suffice for the low fidelity model, but high fidelity state trajectory information will come from actual tracking data. Collecting the online data will take time and will require a good deal of processing to properly condition. In order to demonstrate the method, a simple dynamic model was used to generate data for both the high and low fidelity models. This allows for more flexibility when iterating through step sizes. A simple mass spring damper system will provide the harmonic system for our analysis. Figure 5.1 shows a free body diagram of the system. The model consists of three masses each connected via a spring and damper. There is an input U_i acting on each mass. The datum X_0 is ground fixed while X_i is fixed to each mass. It is assumed that each mass rolls without friction. Table 5.1 shows the constants used in the simulation model. Equation (5.1)

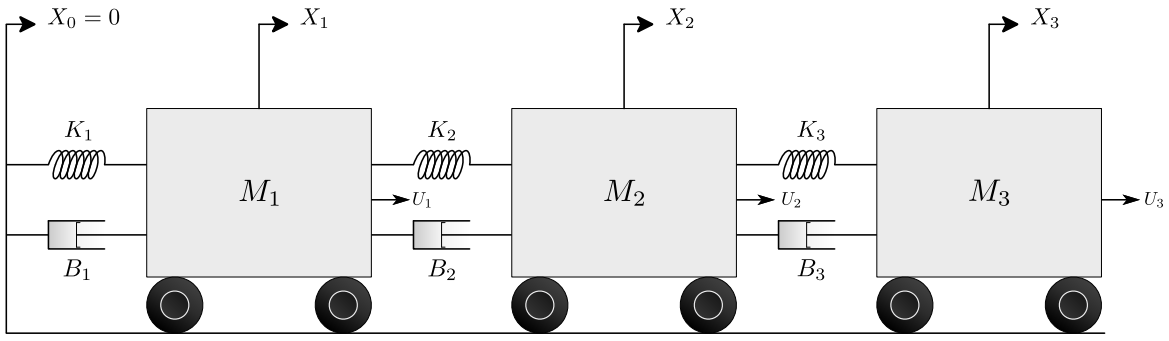


Figure 5.1: Mass Spring Damper System

shows the matrix form of the linear differential equations of motion for the system. This equation is expanded to show the mass-spring-damper system dynamics in Equation (5.2). In order to simulate high and low fidelity state trajectory information, the system inputs are varied. For the low fidelity

Table 5.1: System Constants

$k_1 = 50$	$b_1 = 0.05$	$M_1 = 10$
$k_2 = 25$	$b_2 = 0.05$	$M_2 = 2$
$k_3 = 10$	$b_3 = 0.05$	$M_3 = 2$

data the system inputs U_i are set to zero. A sinusoidal input is applied for the high fidelity state trajectory data. Table 5.2 shows the equations used for input. The system is linear with

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.1}$$

$$\begin{bmatrix} \dot{X}_1 \\ \ddot{X}_1 \\ \dot{X}_2 \\ \ddot{X}_2 \\ \dot{X}_3 \\ \ddot{X}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-(k_1+k_2)}{M_1} & \frac{-(b_1+b_2)}{M_1} & \frac{k_2}{M_1} & \frac{b_2}{M_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_2}{M_2} & \frac{b_2}{M_2} & \frac{-(k_2+k_3)}{M_2} & \frac{-(b_2+b_3)}{M_2} & \frac{k_3}{M_2} & \frac{b_2}{M_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{M_3} & \frac{b_2}{M_3} & \frac{-k_3}{M_3} & \frac{-b_3}{M_3} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ \dot{X}_1 \\ X_2 \\ \dot{X}_2 \\ X_3 \\ \dot{X}_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{M_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{M_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{M_3} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ U_1 \\ 0 \\ U_2 \\ 0 \\ U_3 \end{bmatrix}. \tag{5.2}$$

Table 5.2: Inputs

Low Fidelity	High Fidelity
$U_1 = 0$	$U_1 = 0$
$U_2 = 0$	$U_2 = 15\sin(2\pi t)$
$U_3 = 0$	$U_3 = 20\sin(2\pi t)$

The system was numerically integrated to produce time series state outputs for both high and low fidelity models. The times series data starts at $t = 0$ and ends at $t = 80$s. Figure 5.2 shows the low and high fidelity state output data where m is for meters and s is for seconds.

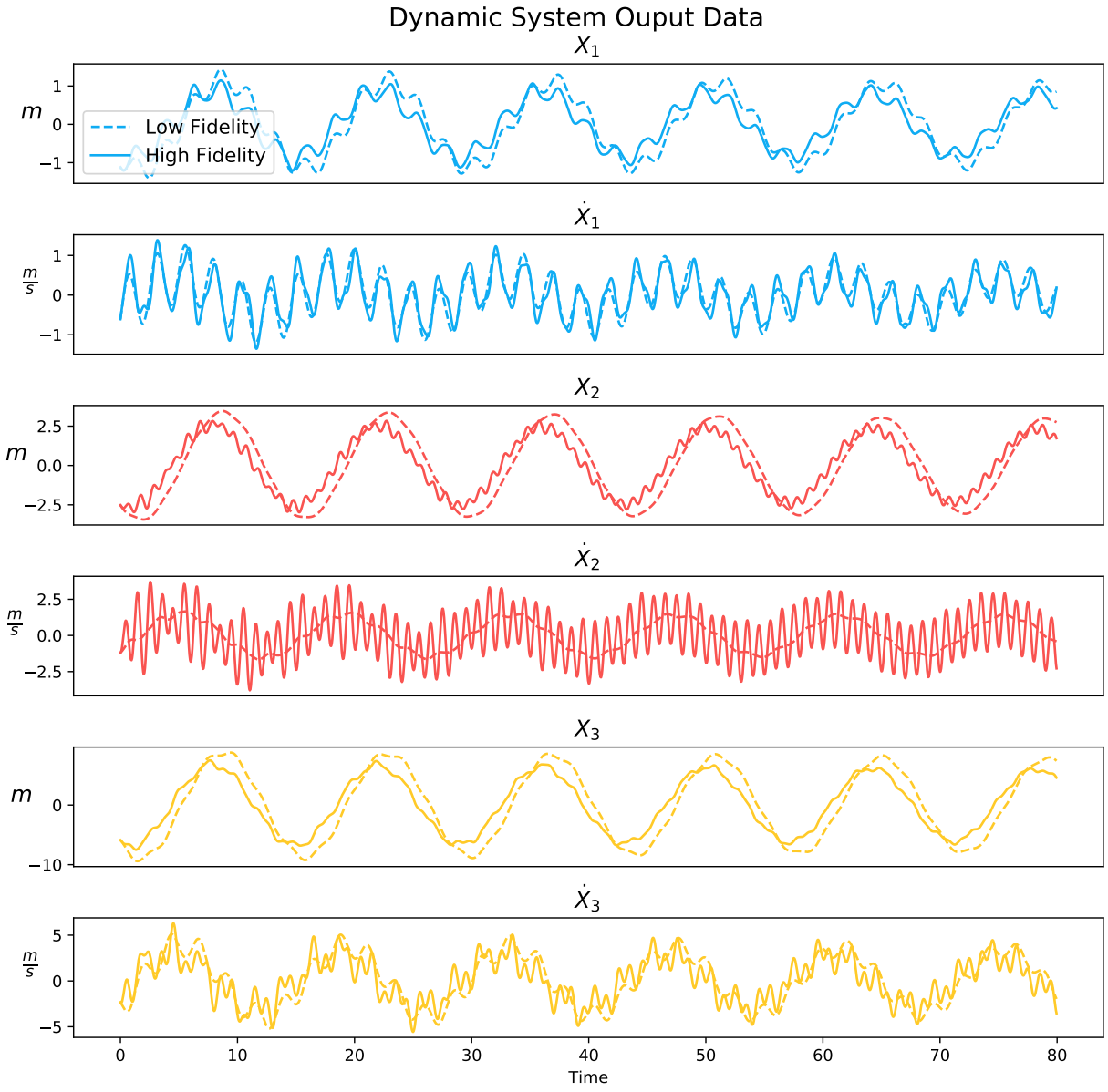


Figure 5.2: Dynamic System State Trajectories

CHAPTER 6

SVD ANALYSIS

In order to develop the training set data, the initial condition step size for each state must be calculated. As discussed before there are two approaches for calculating step size that will be considered. One will consider the variance of each state output in isolation and the other will consider the variance of combinations of states. The SVD will extract the dominant correlations of the state output data. Analyzing these correlations will provide statistical information about the state components associated with feature-rich patterns in the data [23]. The accumulation of information within each pattern represents the data energy of the matrix. This analysis will be performed on the high fidelity state output data. It is assumed that this data exhibits all of the characteristics of the real-world system, and therefore, performing the step size analysis on this data will yield the most accurate step size. Once the step size for each approach is calculated, the training data can be generated from the dynamic system model and used to develop an ANN.

State data is organized into the matrix **S** where the columns of **S** are time series samples of each state. Equation (6.1) shows the state output matrix **S** where n is the number of data points in the time series and m is the number of states.

$$\mathbf{S}_{n\times m} = \begin{bmatrix} | & | & & | \\ \mathbf{state}_1 & \mathbf{state}_2 & \cdots & \mathbf{state}_m \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_m \\ | & | & & | \end{bmatrix} \tag{6.1}$$

Before performing a statistical analysis on the data, it is normalized. Normalizing the data will help ensure the output magnitude does not bias the analysis. First, the mean is taken of each column as shown in Equation (6.2). The column mean is then vectorized in Equation (6.3) to perform matrix

subtraction from matrix **S**.

$$\bar{s}_j = \frac{1}{n}\sum_{l=1}^{n} \mathbf{S}_{lj} \tag{6.2}$$

$$\bar{\mathbf{s}}_j = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{s}_j \tag{6.3}$$

Equation (6.4) shows the data centered matrix **D**.

$$\mathbf{D}_{n\times m} = \begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_m \\ | & | & & | \end{bmatrix} \text{ where: } \mathbf{d}_j = \mathbf{s}_j - \bar{\mathbf{s}}_j \tag{6.4}$$

Next the absolute maximum element from each column of **D** is taken and assembled in vector **q** shown in Equation (6.5).

$$\mathbf{q}_{\,1\times m} = \begin{bmatrix} max(|\mathbf{d}_1|) & max(|\mathbf{d}_2|) & \cdots & max(|\mathbf{d}_m|) \end{bmatrix} \tag{6.5}$$

Once the vector **q** has been calculated, its components are used to scale the columns of matrix **D** by dividing every element in the j^{th} column of **D** by the j^{th} component of vector **q**. Shown in Equation (6.6), matrix **B** contains the state output data that is centered and scaled such that the column-wise means of **B** is zero and the absolute max value of any column element is 1.

$$\mathbf{B}_{n\times m} = \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \\ | & | & & | \end{bmatrix} \text{ where: } \mathbf{b}_j = \frac{1}{q_j}\mathbf{d}_j \tag{6.6}$$

As previously discussed, a simple method for calculating the IC step sizes would be based on the standard deviation on the columns of **B**. However, this will not account for the state covariance. Developed from this research, Equation (6.7) shows the step size calculation using the *Standard Deviation Method* (SDM)

$$SDM_{State_j} = q_j\Big(1 - std(\mathbf{b}_j)\Big) \tag{6.7}$$

where $\mathbf{q}$ is given by Equation (6.5). The step size metric for state s_i is calculated by taking the standard deviation of the normalized time series data $\mathbf{b}_j$. To calculate the step size using the SDM the step size metric is subtracted from 1 and then scaled back to its original magnitude.

A multivariate standard deviation analysis is performed by calculating the SVD of the normalized data in matrix $\mathbf{B}$. Note that the approach shown here is a deviation from the SVD example shown previously. Equation (6.8) shows the SVD calculation for $\frac{\mathbf{B}}{\sqrt{n-1}}$. This will give similar singular values. The SVD of $\frac{\mathbf{B}}{\sqrt{n-1}}$ will yield the square root of the singular values calculated with the eigenvalue decomposition of the covariance matrix of $\mathbf{B}$. The division by $\sqrt{n-1}$ is necessary for calculating the standard deviation of the projected data in the alternate coordinate system. The matrix $\mathbf{V}$ is the same for both. Additionally, calculating the SVD will also yield the matrix $\mathbf{U}$ where the columns of U are the time series modes of the normalized state output matrix $\mathbf{B}$. The matrix dimensions shown here are in the truncated SVD form. The diagonal components of Σ are organized such that σ_1 corresponds to the largest singular value, and so on. Equations (6.8) and (6.9) show the SVD in truncated form with the corresponding matrix dimensions of $\mathbf{U}$, *the left singular vectors of* $\mathbf{B}$, the diagonal matrix Σ, *the singular values of* $\mathbf{B}$, and $\mathbf{V}$, *the right singular vectors of* $\mathbf{B}$. Equation (6.10) shows the SVD equation in expanded form

$$\frac{\mathbf{B}}{\sqrt{n-1}} = \mathbf{U}\Sigma\mathbf{V}^* \textit{ where: } \mathbf{U}_{n\times m}\ \ \Sigma_{m\times m}\ \ \mathbf{V}_{m\times m} \tag{6.8}$$

$$\frac{\mathbf{B}}{\sqrt{n-1}} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_m \\ | & | & & | \end{bmatrix}^* \tag{6.9}$$

$$\frac{\mathbf{B}}{\sqrt{n-1}} = \sum_{k=1}^{m} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_m \mathbf{u}_m \mathbf{v}_m^* \tag{6.10}$$

where * denotes the matrix transpose.

As seen in the previous example, the columns of $\mathbf{V}$ are unit vectors in state space. Each vector $\mathbf{v}_j$ defines a line that spans state space and corresponds with the singular value σ_j. The scalar

value σ_j is the standard deviation of all data projected onto that line. This information provides a covariance analysis of the states. The constituents of $\mathbf{v}_j$ represents the amount that each state is contributing to σ_j. The state components of $\mathbf{v}_j$ will be used to calculate the step size for each state. Keeping in mind that the magnitude of $\mathbf{v}_j$ is 1, Equation (6.11) shows that the sum of the squared components of $\mathbf{v}_j$ is equal to 1.

$$\| \mathbf{v}_j \|^2 = v_{1j}^2 + v_{2j}^2 + \cdots + v_{mj}^2 = 1^2 = 1 \tag{6.11}$$

Equation (6.12) shows that the sum of the squared components of $\mathbf{v}_j$ multiplied by σ_j is equal to σ_j.

$$\sigma_j(v_{1j}^2 + v_{2j}^2 + \cdots + v_{mj}^2) = \sigma_j v_{1j}^2 + \sigma_j v_{2j}^2 + \cdots + \sigma_j v_{mj}^2 = \sigma_j \tag{6.12}$$

Next the matrix $\mathbf{M}$ is assembled as shown in Equation (6.13). As demonstrated in Equation (6.12), Equation (6.14) shows that the sum of the elements of $\mathbf{m}_j$ are equal to σ_j.

$$\mathbf{M}_{m\times m} = \begin{bmatrix} | & | & & | \\ \mathbf{m}_1 & \mathbf{m}_2 & \cdots & \mathbf{m}_m \\ | & | & & | \end{bmatrix} \tag{6.13}$$

$$\mathbf{m}_j = \sigma_j \begin{bmatrix} {v_{1j}}^2 & {v_{2j}}^2 & \cdots & {v_{mj}}^2 \end{bmatrix}^T \text{ where: } \sum_{i=1}^{m} m_{ij} = \sigma_j \tag{6.14}$$

Each vector $\mathbf{m}_j$ corresponds to a dominant correlation within the data and each element of $\mathbf{m}_j$ corresponds to a state. To better visualize this, Table 6.1 displays matrix $\mathbf{M}$ with each row labeled with the corresponding state variable and each column labeled with the corresponding singular value for the example mass-spring-damper system. In order to find the step size metric for state

Table 6.1: SVD State Components ($m = 6$)

State		σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
s_1	X_1	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}
s_2	$\dot{X}_1$	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}	m_{26}
s_3	X_2	m_{31}	m_{32}	m_{33}	m_{34}	m_{35}	m_{36}
s_4	$\dot{X}_2$	m_{41}	m_{42}	m_{43}	m_{44}	m_{45}	m_{46}
s_5	X_3	m_{51}	m_{52}	m_{53}	m_{54}	m_{55}	m_{56}
s_6	$\dot{X}_3$	m_{61}	m_{62}	m_{63}	m_{64}	m_{65}	m_{66}

s_i, the row $\mathbf{m}_i$ is searched for the largest element of any data correlations calculated by the SVD. Developed from this research, Equation (6.15) shows the step size calculation using the *Linear Combination Method* (LCM)

$$LCM_{State_j} = q_j\left(1 - max\left(\begin{bmatrix} m_{i_1} & m_{i_2} & \cdots & m_{i_m} \end{bmatrix}\right)\right) \tag{6.15}$$

where $\mathbf{q}$ is given by Equation (6.5). The larger the step size metric found in $\mathbf{m}_i$ the more data energy is dependent on s_i. Therefore, the more dependence the overall data energy has on s_i, the smaller the step size should be. To calculate the step size using the LCM the step size metric is subtracted from 1 and then scaled back to its original magnitude. Subtracting from 1 will yield a smaller step size when the data energy is more dependent on s_j.

Table 6.2 shows the step sizes for both methods calculated using state trajectories propagated from the initial condition shown. Figure 6.1 shows a bar plot of the step size values from

Table 6.2: Initial Condition Data

	X_1	$\dot{X}_1$	X_2	$\dot{X}_2$	X_3	$\dot{X}_3$
Nominal IC	−1.118	−0.608	−2.515	−1.194	−5.858	−2.318
SDM Step	0.580	0.834	1.199	2.236	2.913	3.830
LCM Step	0.897	1.189	1.495	3.460	2.244	4.756

Table 6.2. This shows the relative difference of corresponding state step sizes between methods. It can be seen that, while some states have similar relative magnitudes, others are completely different. These differences are a result of covariance in the data. In general, larger step sizes are preferred, as they reduce the amount of data in the training set, provided the accuracy is preserved. The singular values plot in Figure 6.2 show the singular values from greatest to smallest. The total energy plot shown in Figure 6.2 shows the cumulative sum of each singular value divided by the total sum. This displays the data energy totals of successive combinations of singular values. For example, calculating the matrix $\frac{\mathbf{B}}{\sqrt{n-1}}$ using only the first two singular values would, as shown in Equation (6.10) where $m = 2$, reconstruct approximately 65% of the original data. Figure 6.3 shows the times series modes of the left singular vectors from matrix $\mathbf{U}$.

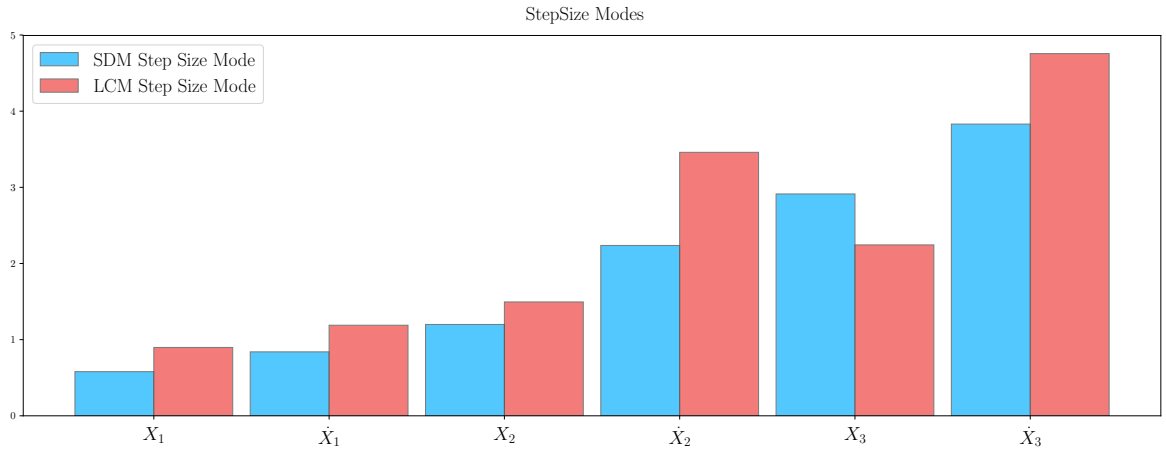


Figure 6.1: Step Size Modes

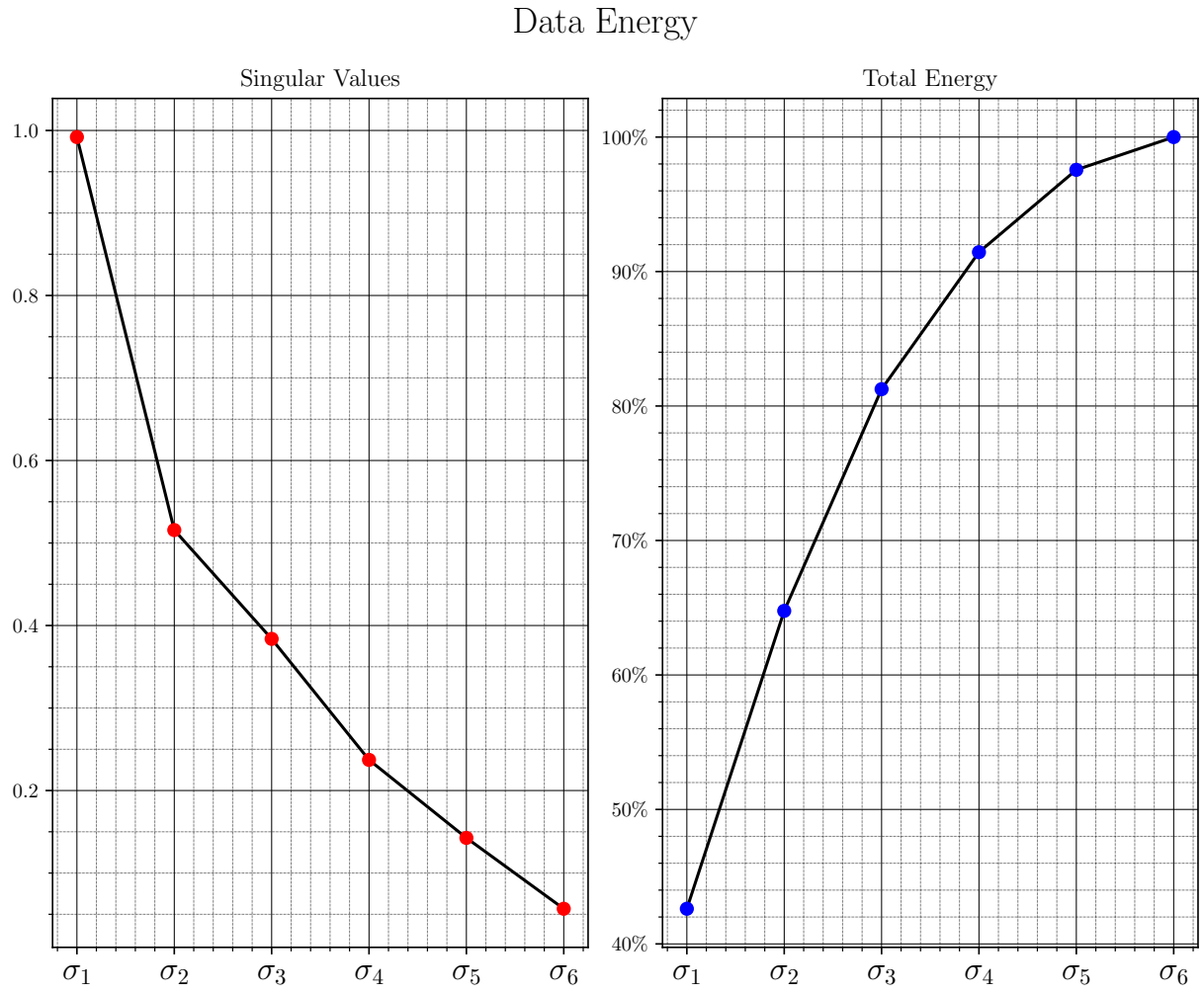


Figure 6.2: Data Energy Analysis

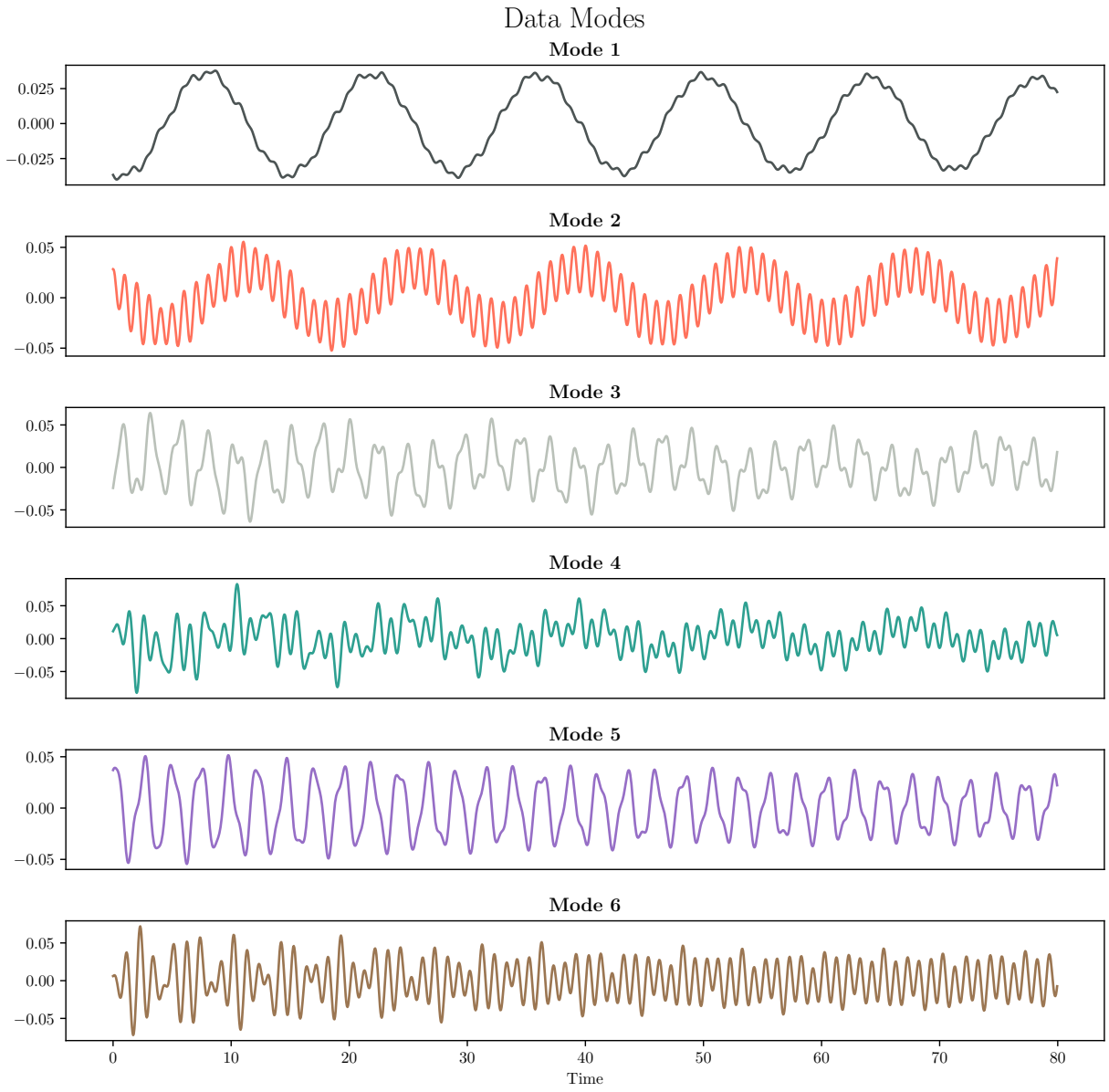


Figure 6.3: Time Series Data Modes (Columns of **U**)

CHAPTER 7

GENERATING TRAINING DATA

The data used to perform the step size analysis is used as the nominal state trajectory data when testing the ANN. The initial condition that created this nominal data is used as the test point on the initial condition grid. The ANN is trained on data generated from initial conditions surrounding the nominal initial condition. The step size between the nominal initial condition and training initial condition is based on the LCM and SDM step sizes calculated previously. This will force the ANN to recreate a state trajectory on which it was not trained. The accuracy with which it recreates the nominal data is one metric by which the two step size methods are measured.

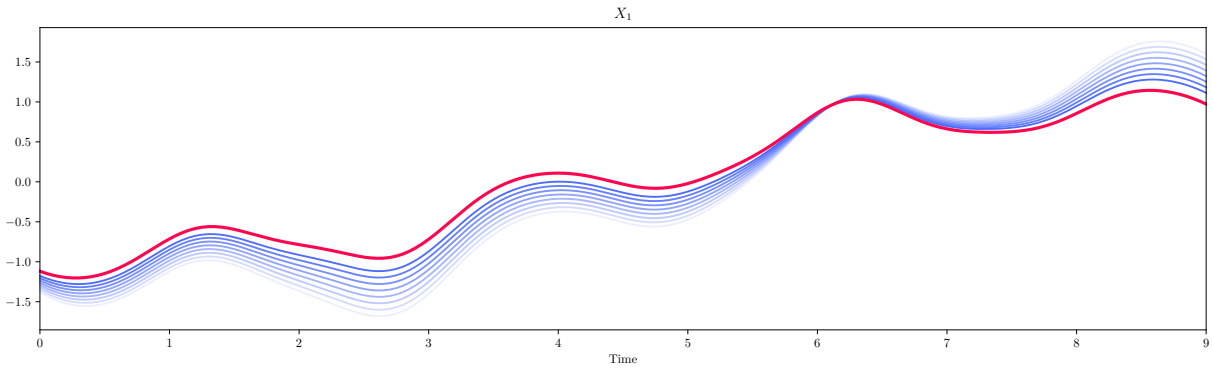
The ANN is trained on every combination of state initial conditions around the nominal IC. There are 6 states for this system giving $2^6 = 32$ initial condition combinations. Table 7.1 shows an example of the initial condition combinations for a three state system in terms of step high and step low from the nominal initial condition. These correspond to vertices on a 3-dimensional cube. 'High' and 'Low' refer to the relative position of the vertices. From each initial condition a full time

Table 7.1: Initial Condition Combinations

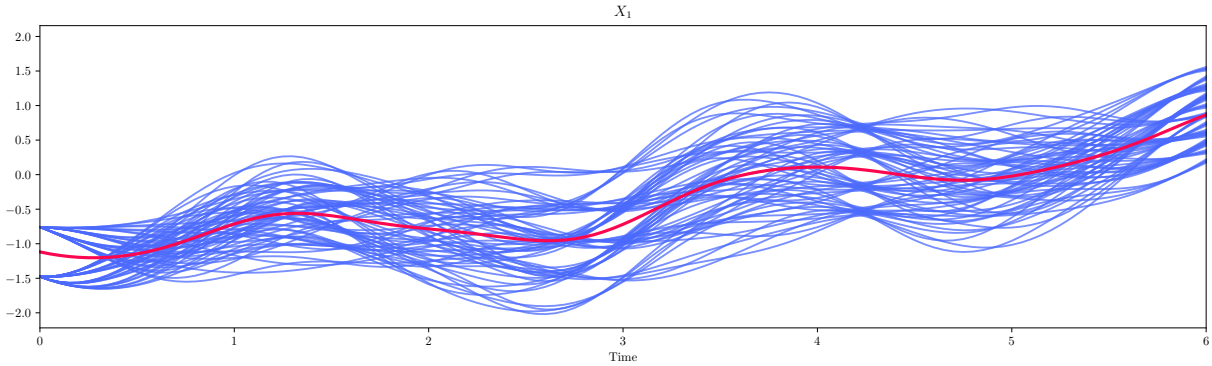
IC 1	High	High	High
IC 2	High	High	Low
IC 3	High	Low	High
IC 4	High	Low	Low
IC 5	Low	High	High
IC 6	Low	High	Low
IC 7	Low	Low	High
IC 8	Low	Low	Low

series state trajectory will be generated from the dynamic system to train the ANN. To test each method, multiple ANNs will be trained with data assembled from varying the overall magnitude of each step size mode. Starting with a reduced magnitude for all step sizes, the magnitude is

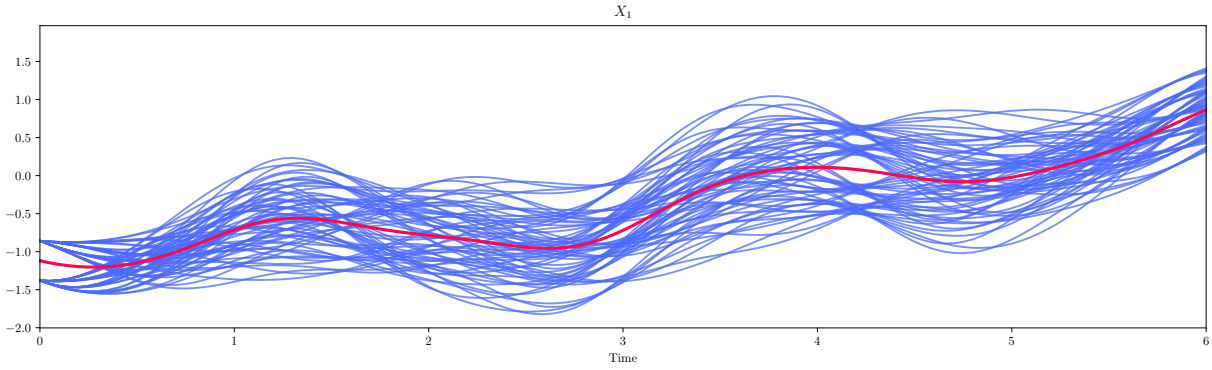
incrementally increased thus sweeping outward from the nominal IC.



(a) SDM Step Size Sweep



(b) LCM Initial Condition Combinations



(c) SDM Initial Condition Combinations

Figure 7.1: Training Data

In Figure 7.1, the red curve in all three plots is the nominal state trajectory X_1. Figure 7.1a shows the X_1 state trajectory data with the initial condition step size swept out from the nominal initial condition. This displays the difference in training data between the ANNs tested. Figures 7.1b and 7.1c show the state trajectory data generated from all initial condition combinations for X_1 for a single mode step size.

CHAPTER 8

ARTIFICIAL NEURAL NETWORK ARCHITECTURE

The ANN architecture used is comprised of six fully connected layers. Low fidelity state data X_{j_L} is input into the network along with its derivative and the time t. High fidelity state data X_{j_H} is the network output. There are ten input neurons in the first layer. Six inputs are for each state of the system, one input is used for the elapsed time where $t = 0$ at the initial condition, and three inputs are for the accelerations of X_1, X_2, and X_3. It was found that the ANN works best when the derivatives of the time series inputs were included. This provides the ANN with information about where each input is headed. Since $\dot{X}_j$ is already included in the training data, only $\ddot{X}_j$ is added. $\ddot{X}_j$ data is generated during the training routine by numerically differentiating the $\dot{X}_j$ data. Figure 8.1 shows the network architecture. The number of neurons in hidden layers 1 through 4 are 200, 1000, 800, and 400, respectively.

All four hidden layers are activated with ReLU activation functions. The ReLU activation function, $a(z) = max(0, z)$, is computationally inexpensive to compute and therefore reduces the time require to update the network when training [24]. The Adam Optimizer [25] was used in back propagation for training the ANN. The training performance loss was measured by Mean Squared Error (MSE) and the validation performance loss was measured by Mean Absolute Error. The training data was randomly reordered with 80% of the data allocated for training and 20% allocated for validation. A limit of 100 epochs was set for training. One epoch is when the training routine has trained on all of the data. After each epoch the network parameters are updated and the training data is again run through the ANN for further training. Also, in the event that the network performance did not improve after 10 epochs, the training routine was stopped and the parameters for the best network were saved.

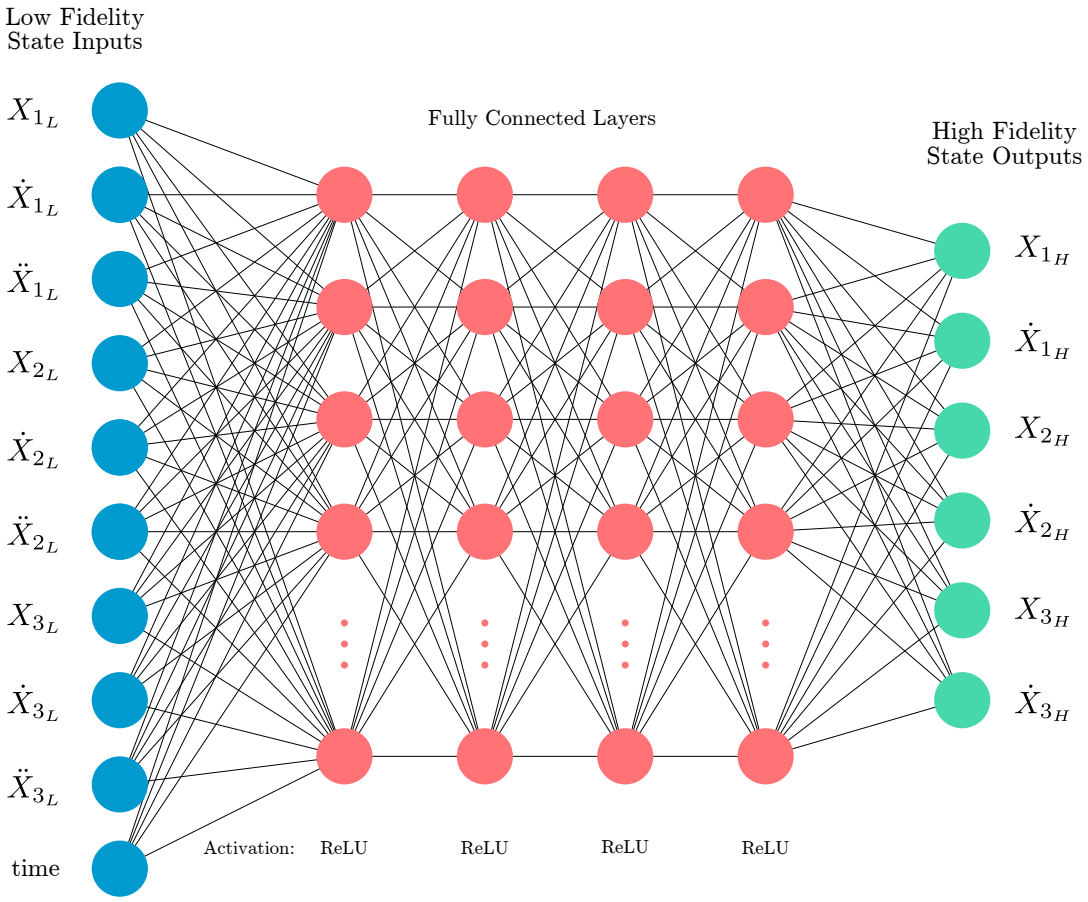


Figure 8.1: Neural Network Architecture

CHAPTER 9

RESULTS

Figure 9.1 shows the output of an ANN trained using the LCM method. The desired nominal high fidelity state trajectory data is plotted on the same plot as the network output. Note that the results from the ANN are virtually indistinguishable from the given data. The mean squared error is approximately 0.00609 between the generated nominal data and the network output. Here the network accuracy is very good but the step size is small and therefore the computation cost for training is higher. In order to measure the high fidelity output accuracy between step size methods,

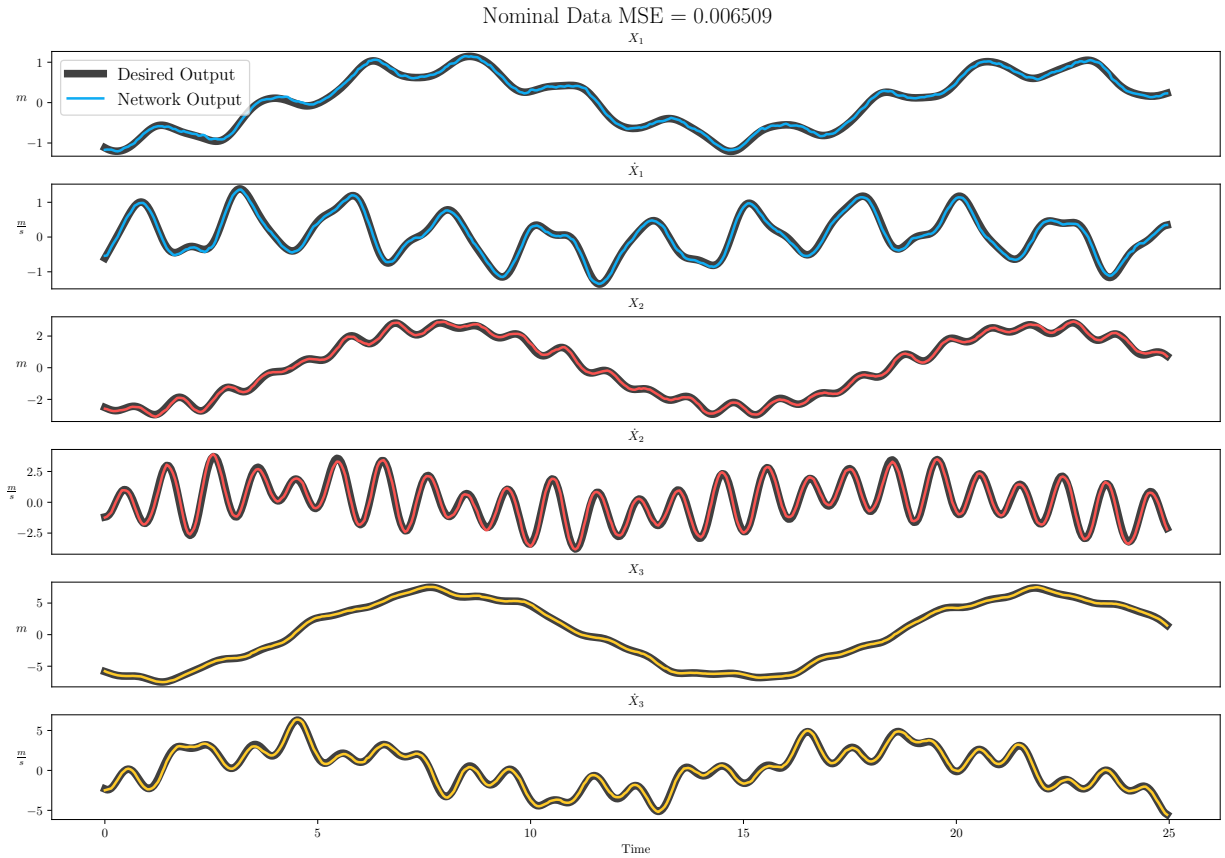


Figure 9.1: LCM Training Routine Best Trained Network

a Step Size Index (SSI), defined by Equation (9.1), was used. This makes the assumption that the valid range for each state is 100 regardless of units or system constraints. Since the goal is to reduce the number of initial conditions needed for training, a greater step size is desirable. The lower the SSI, the larger the overall step size across all state initial conditions.

$$Step\ Size\ Index = \sum_{i=1}^{m} \frac{100}{2\,Step_{s_i}} \tag{9.1}$$

While the step size plays a large role in the computational cost for training the ANN, the number of training epochs is also important. For very large networks, one epoch can take a considerable amount of time. Reducing the number of epochs is desirable as it will require less computation to train the ANN.

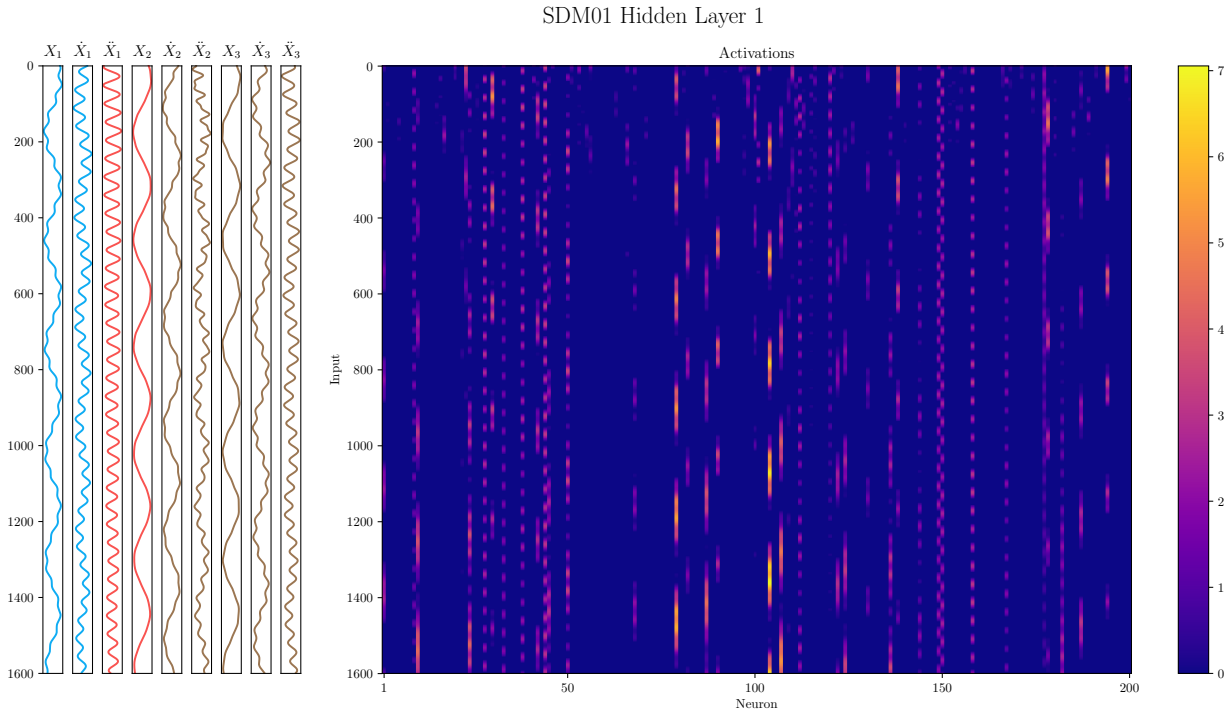
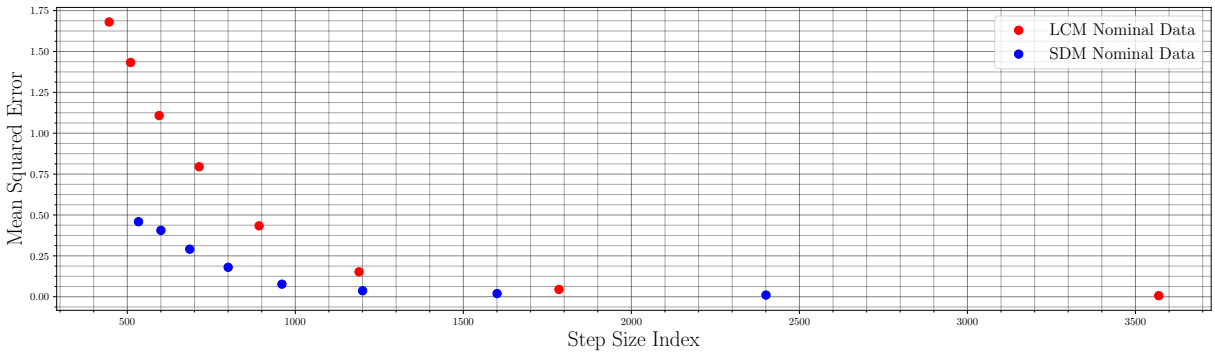


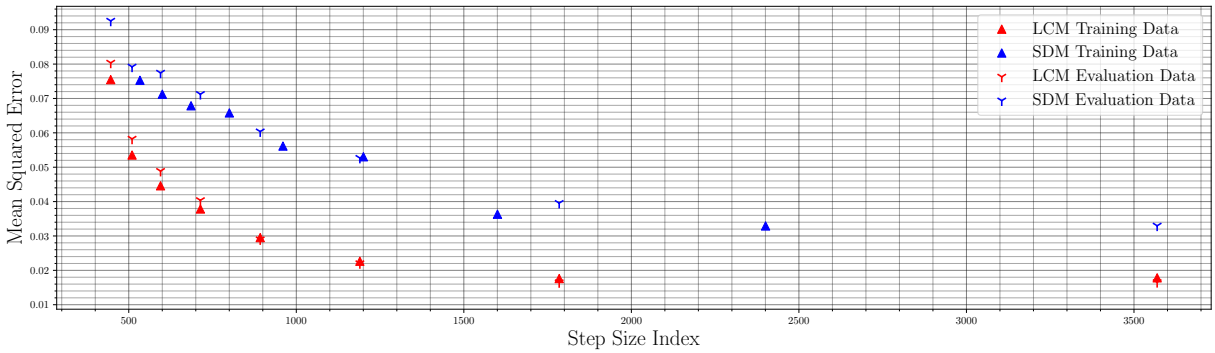
Figure 9.2: Time Series Neural Activations in First Layer

Figure 9.2 shows the time series neural activation for the first layer. The harmonic nature of this system can be clearly seen in the cyclic activation. Figure 9.3a and 9.3b show the network performance against the SSI for the nominal data and training-evaluation data respectively. Figure 9.3a shows the network MSE performance when tested with the nominal data. Note that the ANN was not trained on this data, therefore, this is information that the network must interpolate to pro-

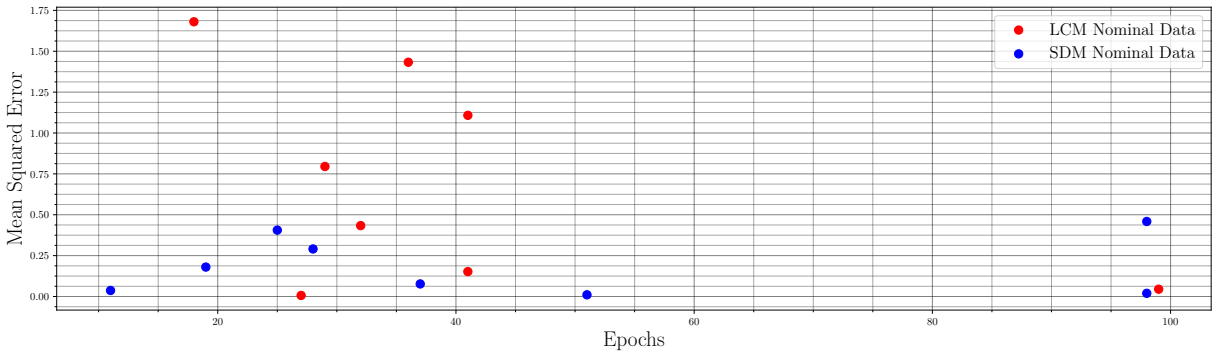
duce. At lower SSI the SDM achieved better results, but at higher SSI the performance between the two methods becomes more evenly matched. It should be noted that a MSE of at least 0.08 was necessary to produce reasonable results. Therefore, much of the lower range of the SSI produced poor performance. Figure 9.3b shows the network MSE performance when tested with the training and validation data. Here the data shows that the LCM achieved better performance with respect to the training and validation data. This suggests that the LCM achieves better performance with respect to the training set data. This may be desirable because while local performance is important, global performance plays a greater role in overall network performance. Figure 9.3c and 9.3d show the network performance against the epochs required for training. Figure 9.3c shows the network MSE performance when tested with the nominal data. The SDM and LCM achieved good performance with relatively few training epochs. Note that the LCM produced the best performing network at 34 epochs. Figure 9.3b shows the network MSE performance when tested with the training and validation data. The LCM produced much better results in fewer epochs for multiple training cycles. Again, the LCM produced the best performing network.



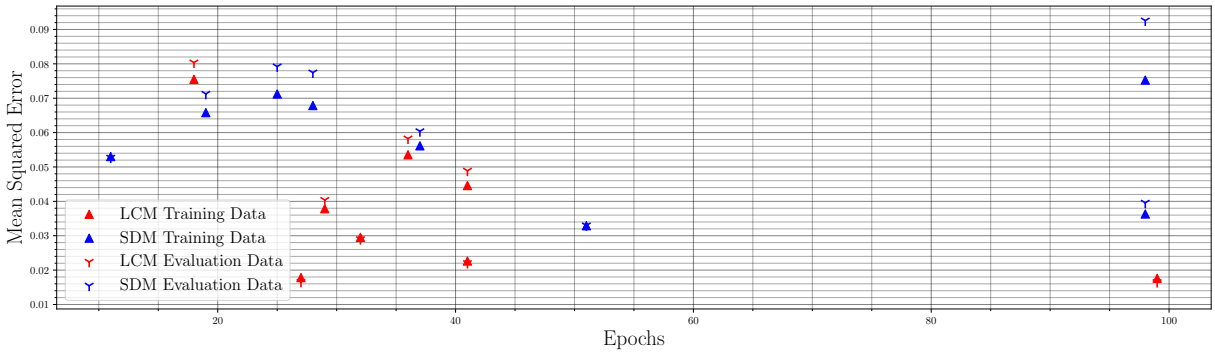
(a) Nominal Data Step Size Index



(b) Training-Evaluation Data Step Size Index



(c) Nominal Data Epochs



(d) Training-Evaluation Data Epochs

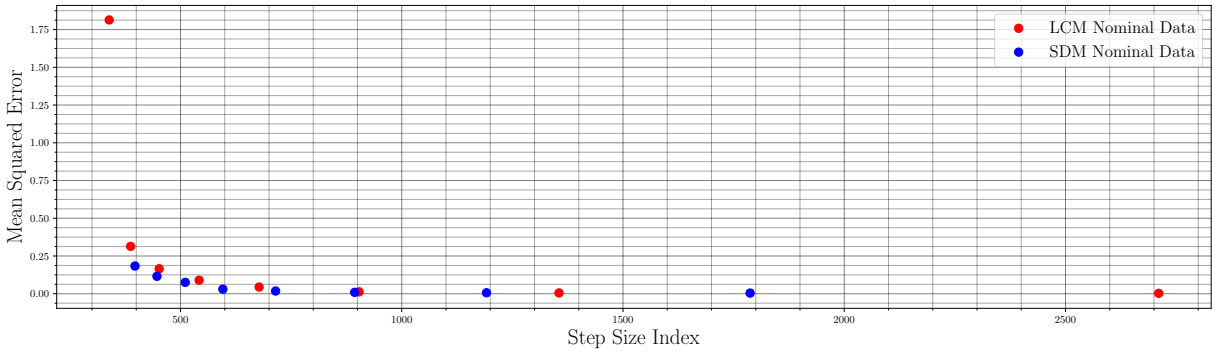
Figure 9.3: ANN Performance from I.C. 1

An additional test was run using a different nominal initial condition. Table 9.1 shows the nominal initial condition used for the second analysis. Figure 9.4a and 9.4b show the second

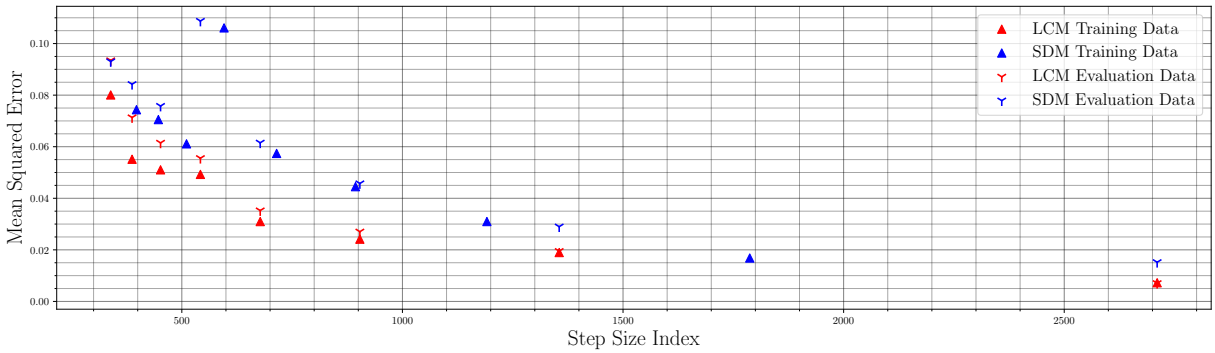
Table 9.1: Initial Condition Data from Second Analysis

	X_1	$\dot{X}_1$	X_2	$\dot{X}_2$	X_3	$\dot{X}_3$
Nominal IC	-0.023	-3.3618	-1.044	-1.203	-2.684	3.921
SDM Step	0.902	1.813	1.007	3.096	2.726	4.319
LCM Step	1.159	2.425	1.443	3.180	3.693	6.348

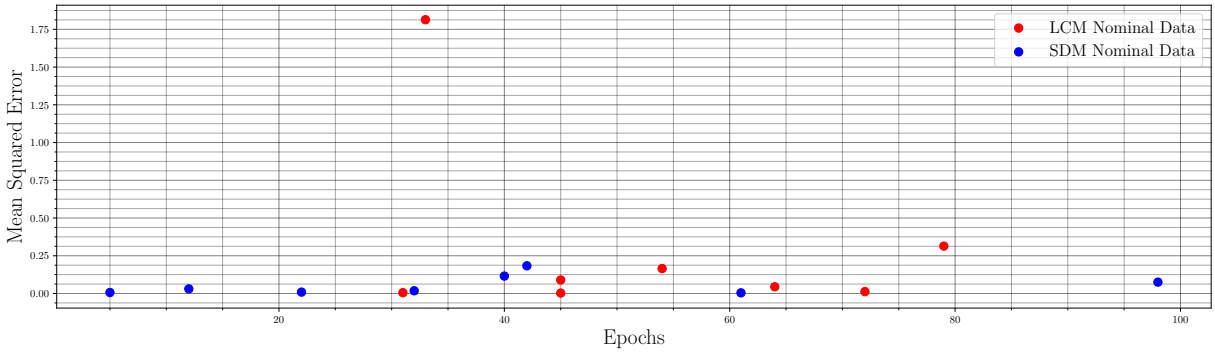
analysis network performance against the SSI for the nominal data and training-evaluation data respectively. Figure 9.4a shows the performance between the two methods is similar when compared the nominal data. Similar to the first analysis, 9.4b shows that the LCM again achieves better performance with respect to the training and evaluation data, again producing the best performing network. Figure 9.4c and 9.4d show the second analysis network performance against the epochs required for training. Figure 9.4c shows that the SDM achieved better performance in fewer epochs while the LCM takes more epochs to achieve the same performance. Figure 9.4d shows that the LCM achieved the best network overall performance.



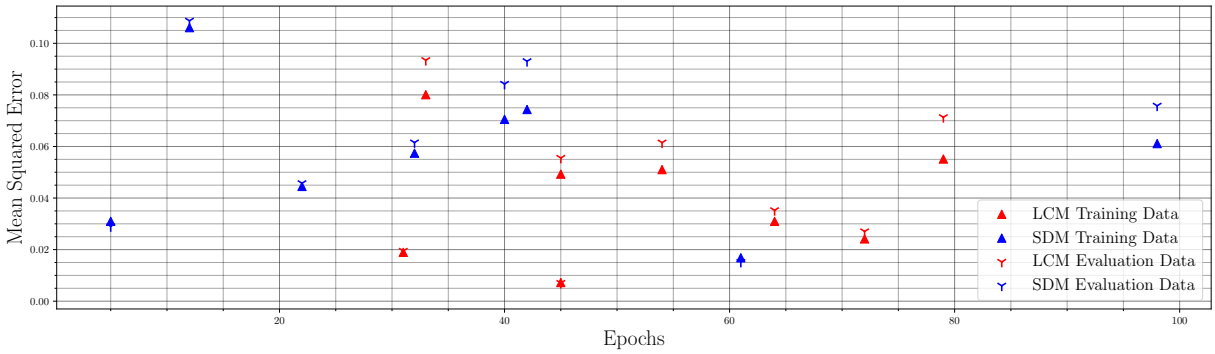
(a) Nominal Data Step Size Index



(b) Training-Evaluation Data Step Size Index



(c) Nominal Data Epochs



(d) Training-Evaluation Data Epochs

Figure 9.4: ANN Performance from I.C. 2

CHAPTER 10

CONCLUSION

Reducing training set density can have a significant impact in the time required to train a ANN. The training set density can only be reduced by eliminating redundant information from the training data. Using a SVD analysis of state data, a method for calculating the initial condition step size to produce example training trajectories has been developed that preserves the data that characterizes the system dynamics.

The examples indicate that the LCM achieves better performance with respect to the entire training set. This may be due to the fact that the step size mode was calculated based on the dominant correlations or patterns within the data set. The LCM used for setting training data density and the ANN architecture tested here show the potential for use in predicting state trajectories for systems represented by low fidelity models and subject to unmodeled perturbations. Orbital state prediction is one application where efficiency in ANN training and reduction of training data density could make a significant impact. More work is needed to test these methods with systems exhibiting non-periodic non-linearity. Since the investigated system was linear, future work should also seek to compare simulation results with analytical solutions based on classical linear system theory.

Due to time and resource constraints, this work was not applied to orbital trajectories after the methods were developed using the example linear system. The next step is to gather real satellite data from a data base. From this data, a statistical analysis, using SVD, could be performed for training data selection and an appropriate low fidelity model could be developed. Algorithms for extracting training data from the gathered data has yet to be developed. Further ANN architecture development is needed to scale this application to a larger data set.

REFERENCES

[1] USAF Scientific Advisory Board. Space surveillance, asteroids and comets, and space debris, volume 1, space surveillance. Technical report, SAB-TR-96-04, June, 1997.

[2] USAF; Maj Edward P. Chatters IV and USAF Maj Brian J. Crothers. *AU-18 Space Primer*. Air University Press, Maxwell Air Force Base, Ala, 2009. Chapter 19: Space Surveillance Network.

[3] Michael W Taylor. Orbital debris: Technical and legal issues and solutions. Technical report, MCGILL UNIV MONTREAL (QUEBEC), 2006.

[4] Planet.com. Flock 2k blog posts. Technical report. `https://www.planet.com/pulse/tag/flock-2k/`, Accessed March 2019.

[5] David Vallado and Paul Cefola. Two-line element sets - practice and use. *Proceedings of the International Astronautical Congress, IAC*, 7:5812–5825, 01 2012.

[6] Space-track. Technical report. `https://www.space-track.org/`, Accessed March 2019.

[7] Kyle J DeMars, Islam I Hussein, Carolin Frueh, Moriba K Jah, and R Scott Erwin. Multiple-object space surveillance tracking using finite-set statistics. *Journal of Guidance, Control, and Dynamics*, 38(9):1741–1756, 2015.

[8] Islam I Hussein, Kyle J DeMars, Carolin Früh, Richard S Erwin, and Moriba K Jah. An aegis-fisst integrated detection and tracking approach to space situational awareness. In *2012 15th International Conference on Information Fusion*, pages 2065–2072. IEEE, 2012.

[9] Emmanuel Delande, Jérémie Houssineau, and Moriba Jah. Physics and human-based information fusion for improved resident space object tracking. *Advances in Space Research*, 62(7):1800–1812, 2018.

[10] Daniel P Lubey and Daniel J Scheeres. Supplementing state and dynamics estimation with information from optimal control policies. In *17th International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2014.

[11] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *CoRR*, abs/1603.08029, 2016.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Richard Linares and Roberto Furfaro. An autonomous sensor tasking approach for large scale space object cataloging. In *Proc. the Advanced Maui Optical and Space Surveillance Technologies Conf.(AMOS)*, 2017.

[14] Howard D. Curtis. *Orbital Mechanics for Mechanical Students*. Elsevier Butterworth-Heinemann, 30 Corporate Drive, Burlington, MA 01803, 2005. Chapter 2: Two Body Problem.

[15] Archie E Roy. *Orbital motion*. CRC Press, 2004.

[16] Harry C Koons and David J Gorney. A neural network model of the relativistic electron flux at geosynchronous orbit. *Journal of Geophysical Research: Space Physics*, 96(A4):5549–5556, 1991.

[17] David Pérez, Brendt Wohlberg, Thomas Alan Lovell, Michael Shoemaker, and Riccardo Bevilacqua. Orbit-centered atmospheric density prediction using artificial neural networks. *Acta Astronautica*, 98:9–23, 2014.

[18] KENNETHE WILLIAMS. Prediction of solar activity with a neural network and its effect on orbit prediction. *Johns Hopkins APL Technical Digest*, 12(4):310–317, 1991.

[19] KP Macpherson, AJ Conway, and JC Brown. Prediction of solar and geomagnetic activity data using neural networks. *Journal of Geophysical Research: Space Physics*, 100(A11):21735–21744, 1995.

[20] Hao Peng and Xiaoli Bai. Using artificial neural network in machine learning approach to improve orbit prediction accuracy. *2018 Space Flight Mechanics Meeting*, Jul 2018.

[21] Coryn AL Bailer-Jones, David JC MacKay, and Philip J Withers. A recurrent neural network for modelling dynamical systems. *network: computation in neural systems*, 9(4):531–547, 1998.

[22] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[23] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. Dynamic mode decomposition. 2016.

[24] Relu. Technical report. `https://www.tinymind.com/learn/terms/relu`, Accessed March 2019.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.