Intelligent Monitoring of Powerline Vibrations - Sparse Sensing and Predictive Modeling Approach

Nipun Remasan Nambiar

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Oumar Barry, Chair

Erik Komendera

Suyi Li

April 30, 2025

Blacksburg, Virginia

Keywords: Data-Driven Modeling, Dynamic State Estimation, Sparse Sensor Networks, Wind-Induced Vibrations (WIV)

Intelligent Monitoring of Powerline Vibrations - Sparse Sensing and Predictive Modeling Approach

Nipun Remasan Nambiar

(ABSTRACT)

Wind-induced vibrations (WIV), particularly Aeolian vibrations, pose a persistent threat to the structural integrity and operational lifespan of overhead power transmission lines. Traditional mitigation strategies, such as passive dampers, often lack adaptability to changing environmental conditions and fail to provide real-time insights into the dynamic behavior of conductors. This thesis presents a data-driven framework for real-time monitoring and state estimation of vibration profiles in transmission lines, offering a scalable and intelligent alternative to conventional methods.The proposed approach integrates a range of techniques including LSTM-based neural networks for time-series prediction, Dynamic Mode Decomposition (DMD) with Tikhonov regularization for stable reduced-order modeling, and estimation algorithms such as Kalman and particle filters for reconstructing full-system states from sparse observations. A pivoted QR decomposition method is employed to determine optimal sensor placement, ensuring efficient coverage and accurate reconstruction with minimal hardware. Additionally, compressive sensing is introduced to recover missing data, enhancing system robustness under conditions of partial sensor failure or data loss. Simulation results validate the effectiveness of the proposed methods, demonstrating accurate vibration profile reconstruction and reliable antinode tracking with a reduced sensor set. The framework provides a foundation for intelligent asset management in power systems, with implications for vibration mitigation, infrastructure resilience, and smart grid applications.

Intelligent Monitoring of Powerline Vibrations - Sparse Sensing and Predictive Modeling Approach

Nipun Remasan Nambiar

(GENERAL AUDIENCE ABSTRACT)

Overhead power lines are a vital part of our electric grid, carrying electricity over long distances to homes, businesses, and critical infrastructure. However, these lines are constantly exposed to environmental forces like wind, which can cause them to vibrate. Over time, these vibrations—especially a type called Aeolian vibrations—can lead to wear and damage, increasing the risk of failures, costly repairs, or even dangerous outages. This thesis explores a new, intelligent way to monitor and track these vibrations in real time. Instead of relying solely on traditional hardware-based methods, it uses data and machine learning to predict how the lines are moving. With only a few sensors placed in the right spots, the system can accurately estimate what's happening along the entire length of the power line. It also uses techniques from signal processing to fill in missing information when data is lost or incomplete. The goal is to create a smarter, more efficient way to maintain the safety and reliability of power lines—especially as they age and face greater demands. By combining computer models, learning algorithms, and smart sensor strategies, this research offers a path toward more resilient and cost-effective grid infrastructure.

Dedication

To my parents, for your unwavering love, support, and belief in me—this journey would not have been possible without your constant encouragement. To my friends, Thank you for keeping me grounded, motivated, and sane through the highs and lows of this process. Your presence made even the most stressful days bearable. And to my girlfriend, for your patience, understanding, and endless support—you've been my steady light through every late night and long hour. This thesis is for all of you.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Oumar Barry, whose guidance, insight, and support have been pivotal throughout this research journey. His expertise and encouragement pushed me to explore new ideas and challenged me to grow both technically and intellectually. I would also like to sincerely thank Dr. Erik Komendera and Dr. Suyi Li for serving on my committee and for offering their valuable feedback and suggestions, which greatly enriched the quality of this thesis. This work would not have been possible without the unwavering support of my family. To my parents—thank you for your endless love, patience, and belief in me. Your encouragement has been the foundation of all my pursuits. To my friends, thank you for being a constant source of motivation and for reminding me to laugh even on the most stressful days. Special thanks to my girlfriend—for your compassion, patience, and understanding. Your support has been a steady light during long hours and late nights. Lastly, I would like to acknowledge the Virginia Tech community, the Department of Mechanical Engineering, and all those who contributed—directly or indirectly—to the completion of this thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

The electric grid is one of the most critical components of modern infrastructure, facilitating the transmission and distribution of electricity across vast distances from generation sources to end users. This expansive network of high-voltage transmission lines serves as the backbone of national energy security and economic stability. However, with much of the U.S. electric grid aging—70 of transmission lines being over 30 years old and some components exceeding a century in operation—the need for proactive asset management has become more urgent than ever [3]. The American Society of Civil Engineers assigned the energy sector a C- rating in 2021, citing increasing vulnerabilities due to aging infrastructure, growing electricity demand, and the accelerating transition toward renewable energy sources [3]. A Princeton study estimates that to meet projected energy demands by 2050, the U.S. may need to triple its transmission capacity [4]. Such expansion will necessitate significant investments in grid modernization to improve reliability, efficiency, and resilience in the face of evolving challenges.

Among the many challenges facing transmission infrastructure, wind-induced vibrations (WIV) present a persistent and often overlooked threat to the structural integrity of overhead conductors. WIV, particularly Aeolian vibrations, arise due to the periodic shedding of vortices as wind flows across cylindrical conductors. This phenomenon, known as vortex

shedding, generates alternating forces that can excite the conductor into oscillatory motion. When the shedding frequency aligns with the natural frequency of the conductor, resonance can occur, amplifying the vibration amplitude and accelerating fatigue damage. Aeolian vibrations are typically characterized by small amplitudes and high frequencies, ranging from 3 Hz to 150 Hz, and are driven by wind speeds between 1 m/s and 7 m/s [7, 10, 13, 24, 26, 29, 44, 48]. The cumulative effects of sustained high-cycle bending stresses can lead to mechanical wear, abrasion, and ultimately fatigue failure of the conductor at suspension points [9, 25, 26, 34, 44, 45].

Beyond Aeolian vibrations, power lines are also susceptible to other forms of WIV, such as galloping and wake-induced oscillations. Galloping is a low-frequency (1-3 Hz), high-amplitude motion that arises from aerodynamic instabilities in asymmetric ice- or debris-covered conductors. Wake-induced oscillation, on the other hand, occurs in bundled conductors when aerodynamic interference between adjacent sub-conductors induces oscillatory motion at intermediate frequencies. While each type of WIV presents unique challenges, Aeolian vibrations remain particularly problematic due to their high frequency and cumulative fatigue effects over extended periods.

The consequences of unchecked WIV are substantial, both in terms of safety and economic impact. Fatigue-induced failure of conductors can lead to power outages, infrastructure damage, and, in extreme cases, catastrophic wildfires. For instance, the 2018 Camp Fire in California, which resulted in 84 fatalities and billions of dollars in damages, was traced back to the failure of aging transmission equipment [19]. Even minor disruptions can have severe financial repercussions; data centers, for example, incur an average cost of 9,000 per minute of downtime. In addition to direct damages, utilities face mounting operational and maintenance costs, with transmission infrastructure spending increasing from 9.1 billion in 2000 to 40 billion in 2019. Given the increasing reliance on the electric grid—exacerbated

by the proliferation of electric vehicles, renewable energy integration, and climate-driven demand shifts—enhancing the resilience of transmission infrastructure is an urgent priority.

Despite the well-documented risks associated with WIV, current mitigation strategies remain inadequate in effectively preventing fatigue damage to overhead conductors. Traditional countermeasures, such as dampers and spacer dampers, provide partial solutions but are often reactive rather than preventive. Furthermore, inspecting and maintaining transmission lines is both logistically challenging and cost-intensive, particularly in remote or inaccessible locations. These limitations underscore the necessity for innovative approaches to WIV mitigation that leverage advanced modeling, real-time monitoring, and adaptive control strategies.

This thesis focuses on the intelligent monitoring of Aeolian vibrations in single overhead conductors using data-driven modeling and sparse sensing strategies. Instead of relying solely on traditional hardware-based mitigation techniques, this research develops and validates a predictive framework that leverages methods such as Dynamic Mode Decomposition (DMD), Kalman and particle filtering, Long Short-Term Memory (LSTM) networks, and compressive sensing to reconstruct the full dynamic state of powerline vibrations from limited sensor data. By enabling real-time antinode tracking and robust state estimation under partial observability, this work contributes to a scalable and adaptive approach to vibration monitoring. The proposed framework offers new avenues for efficient asset management and proactive maintenance in transmission systems, with broader implications for improving grid reliability, infrastructure resilience, and operational sustainability.

1.2 Present Methodology

1.2.1 Vibration Absorbers and Mitigation Strategies

A widely used approach to mitigating wind-induced vibrations (WIV) in transmission line conductors is the implementation of vibration absorbers, particularly Fixed Passive Vibration Absorbers (FPVAs) and Tuned Mass Dampers (TMDs). These devices function by dissipating vibrational energy, thereby reducing the amplitude of oscillations and limiting conductor fatigue [11, 12, 44, 56, 57]. Among them, the most commonly deployed FPVA for transmission lines is the Stockbridge damper (SD), a type of passive TMD.

TMDs, first conceptualized by Herman Frahm, consist of a secondary mass, a spring, and a viscous damper, all attached to a primary vibrating structure to counteract its motion. Since their inception, TMDs have undergone extensive optimization, resulting in passive, semi-active, and active control mechanisms. Passive TMDs are widely used due to their structural simplicity and inherent stability; however, their effectiveness is limited in applications where broadband disturbances are present. Moreover, degradation of structural parameters over time reduces their long-term efficiency. Active TMDs address some of these limitations by incorporating sensors and actuators that adjust the absorber's response dynamically. While this enhances performance, active dampers are susceptible to control-induced instability. More recently, semi-active TMDs have gained attention as they combine the benefits of both passive and active systems by allowing controlled adjustments of parameters such as absorber stiffness to optimize vibration suppression.

For power transmission lines, the Stockbridge damper, first developed by George H. Stockbridge in 1925, remains the most prevalent passive vibration mitigation device. The original design, known as the symmetric Stockbridge damper or 2R damper, features two counter-

weights positioned symmetrically on either side of a messenger cable, providing two resonant frequencies within the Aeolian vibration range. Modern iterations have replaced the original concrete counterweights with steel, improving durability and performance. The fundamental damping mechanism involves transferring the conductor's vibrational energy into the damper masses, thereby dissipating the energy more effectively.

The effectiveness of Stockbridge dampers depends on two primary factors: their ability to match the excitation frequencies of the conductor and their placement along the transmission line. Because damping performance is maximized at resonant frequencies, advancements in design have sought to increase the number of resonance modes. One significant improvement is the development of asymmetric Stockbridge dampers, which incorporate unequal counterweights and varying messenger cable lengths, enabling them to achieve four resonant frequencies instead of two. Expanding the number of resonance modes enhances damping efficiency across a broader frequency range and extends the fatigue life of both the conductor and the damper itself.

Further advancements have pushed this concept even further. Barry et al recently patented an updated asymmetric damper design capable of covering six to ten resonant frequencies [8]. This enhancement is achieved by modifying the counterweight geometry, with one side of the counterweight being larger than the other. Computational simulations, such as those conducted in SolidWorks, have validated these designs, demonstrating the presence of six distinct resonant modes. This adaptability allows for customized dampers tailored to specific conductor properties, further improving vibration mitigation efficiency.

Despite these improvements, the placement of Stockbridge dampers remains a key challenge. Ideally, these devices should be installed at antinodes, where vibration amplitudes are highest, to maximize energy dissipation. However, because antinode positions shift with wind velocity, ensuring optimal placement across varying wind conditions is practically unfeasi-

ble. While extensive research has been conducted on optimizing damper positioning, no placement strategy guarantees consistent performance across all wind-induced vibration frequencies. In some cases, a damper may be positioned at or near a node—where vibrational energy is minimal—rendering it ineffective or even detrimental. The added mass at a node can increase local stress, potentially accelerating conductor fatigue rather than mitigating vibrations[37, 46].

Given these constraints, the pursuit of more adaptable vibration mitigation solutions remains an ongoing area of research. Recent developments in smart dampers and tunable mass absorbers present promising alternatives, offering the potential for dynamic adaptation to changing wind conditions. Such innovations could significantly enhance the effectiveness of vibration mitigation in power transmission systems, reducing fatigue-related failures and extending the operational lifespan of conductors.

1.2.2 Inspection Robots

The field of inspection robotics is rapidly advancing, with increasing applications across various industries. The market for inspection robots experienced a significant growth of 31.6 from 2020 to 2021, reaching a valuation of 1.237 billion [1]. In the context of power transmission infrastructure, these robots offer the potential to reduce the cost and complexity of mandatory inspections required to monitor transmission lines for wind-induced vibrations (WIV) and other structural hazards. According to PG and E's *Electric Transmission Preventive Maintenance Manual*, routine patrols must be conducted quarterly, with more comprehensive inspections scheduled annually [42]. Traditionally, these assessments have been carried out through ground patrols or aerial surveys via helicopters. However, both methods are labor-intensive, costly, and, in some cases, hazardous.

To address these challenges, various transmission line inspection robots have been developed [2, 19, 36, 40, 43, 54, 60]. Despite their potential benefits, widespread adoption has been limited due to factors such as high costs, significant power requirements, and the considerable weight and size of these systems. For example, the LineSpyX robot weighs approximately 25 kg, posing logistical challenges for deployment and operation [23]. While Unmanned Aerial Vehicles (UAVs) have become increasingly utilized for inspection purposes, they too face several operational constraints, including limited flight duration, minimum clearance requirements, reliance on operator control, and restrictions on beyond-visual-line-of-sight (BVLOS) operation [55, 59]. Furthermore, existing robotic solutions are not designed to incorporate autonomous operation or contribute to vibration mitigation, highlighting a key area for further innovation and development.

1.2.3 Mobile Damping Robot

Recent advances in powerline monitoring and vibration control have introduced the concept of a Mobile Damping Robot (MDR)—an autonomous device capable of navigating along overhead transmission conductors to dynamically suppress wind-induced vibrations (WIV), particularly Aeolian vibrations [9, 22, 58]. Traditional mitigation methods such as Fixed Passive Vibration Absorbers (FPVAs) suffer from placement limitations, as they rely on static positioning that may not align with shifting antinode locations under varying wind conditions. The MDR addresses this issue by adaptively repositioning itself to the real-time locations of vibration antinodes, thereby maximizing damping efficiency and reducing long-term fatigue damage to the conductor[31, 32, 33].

While conceptual models of MDRs have demonstrated potential in simulations, practical deployment requires a deeper understanding of how such a system would interact with the

dynamic behavior of the conductor. The presence of a mobile mass influences local vibration modes, making static analytical models insufficient for guiding real-time MDR positioning. Prior research has often assumed complete knowledge of conductor parameters and excitation inputs—assumptions that are rarely satisfied in real-world conditions.

This thesis contributes to the foundation necessary for implementing mobile damping strategies by developing a robust, data-driven framework capable of reconstructing full vibration profiles from sparse sensor data. By employing techniques such as Dynamic Mode Decomposition (DMD), Kalman filtering, and LSTM-based prediction, this research enables real-time antinode tracking without requiring a priori knowledge of system parameters. The ability to accurately estimate and track the evolving state of the powerline under partial observability makes it possible to inform future control schemes for mobile dampers.

Ultimately, the integration of such intelligent monitoring frameworks with mobile damping technologies could lead to adaptive vibration mitigation solutions that are both responsive and autonomous. This aligns with the broader goals of infrastructure resilience and smart grid modernization, providing a path forward for more proactive and efficient transmission line asset management.

1.3 Motivation

The growing complexity of modern engineering systems has necessitated the evolution of advanced analytical techniques for understanding and managing their dynamics. Traditional methods, rooted in detailed physical modeling and numerical simulations, often struggle to keep pace with the demands of high-dimensional, real-time systems. This gap between theoretical constructs and practical applicability has spurred significant interest in alternative approaches that leverage the power of data. Data-driven modeling, as a paradigm,

offers a compelling solution to these challenges by focusing on the extraction of patterns and dynamics directly from observed data without relying heavily on explicit physical assumptions. The motivation for adopting data-driven methodologies lies in their ability to address the shortcomings of conventional approaches. Classical numerical methods, while rigorous and accurate, are frequently computationally intensive and time-consuming. For example, finite element methods (FEM) and other discretization techniques require extensive preprocessing, including mesh generation, boundary condition specification, and iterative convergence checks. Such processes can become infeasible for systems with rapidly changing dynamics, such as overhead powerlines subjected to wind-induced vibrations (WIV). Additionally, these methods often depend on simplifying assumptions that may fail to capture the true complexity of the system, leading to inaccuracies in predictions. In the context of vibration suppression in powerlines, one critical limitation of traditional methods is their reliance on predefined models that assume fixed structural properties. However, real-world systems are seldom static; they exhibit variability due to environmental factors, material degradation, and other external influences. Fixed models fail to adapt to these changes, making them unsuitable for applications requiring real-time state estimation and control. Furthermore, numerical methods that do attempt to incorporate adaptability—such as iterative updating schemes—introduce additional computational overhead, further detracting from their practicality. To overcome these limitations, data-driven modeling emerges as a transformative approach. By focusing directly on the observed data, these methods can construct low-dimensional representations of high-dimensional systems, capturing their essential dynamics without the need for explicit physical models. Techniques like Dynamic Mode Decomposition (DMD) excel in identifying coherent structures and temporal patterns in data, offering insights into the underlying behavior of complex systems. Moreover, data-driven approaches inherently adapt to changes in system dynamics, providing a robust framework for applications where variability and uncertainty are prevalent. The shift toward data-driven

methodologies also aligns with broader trends in computational science, where the emphasis has moved from deterministic modeling to probabilistic and empirical approaches. The ability of data-driven methods to handle noise, incomplete data, and nonlinearity makes them particularly appealing for engineering applications. For instance, the integration of data-driven modeling with state estimation techniques such as the Kalman Filter further enhances their applicability by enabling real-time tracking of system states, even when only partial observations are available.

Chapter 2

Data Driven Modeling

This chapter delves into the principles and implementation of data-driven modeling, emphasizing its role as a viable alternative to traditional numerical methods. It begins by exploring the mathematical foundations of key techniques, such as Dynamic Mode Decomposition and Tikhonov Regularization, and progresses to discuss their integration into state estimation frameworks like the Kalman Filter. The discussion highlights the advantages of data-driven approaches, including computational efficiency, adaptability, and resilience to uncertainty, setting the stage for their application to the dynamic state tracking of overhead powerlines.

2.1 Mathematical Modeling

A mathematical model of the system was developed and analyzed using MATLAB®. While the motion of a power line conductor can be described using Newton's Second Law, defining a precise mathematical equation for its movement is challenging due to the conductor's complex structure. To simplify this, the conductor was approximated as a solid cylindrical body with a homogeneous cross-sectional structure. The vibration behavior of the conductor was modeled as a simply supported Euler-Bernoulli beam under axial tension, while the Mass-Damping Robot (MDR) was represented as an attached spring-mass-damper system. A schematic representation of this mathematical model is provided in 2.1. In this model, x denotes the horizontal position of the MDR along the conductor span, m_i represents the

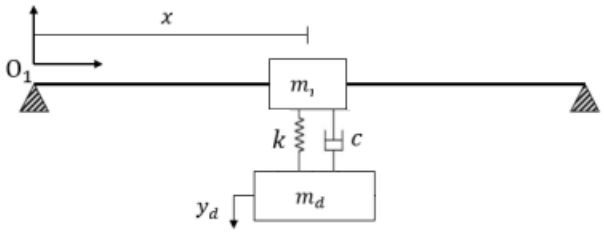


Figure 2.1: Schematic of the power line with MDR

in-span robot mass, k is the spring constant of the absorber, c is the damping coefficient, m_d is the suspended absorber mass, and y_d corresponds to the vertical displacement of the suspended absorber mass. Unlike the work of Kakou et al. [30, 31, 33], the MDR in this study was not modeled as a moving vibration absorber. Instead, the focus was on analyzing the impact of the MDR when stationary at various locations along the cable rather than during traversal. Additionally, energy harvesting was not considered within the scope of this work.

To define the system's dynamics, position vectors for the beam, in-span mass, and suspended mass were first established to describe their respective positions and displacements. Time derivatives of these vectors were then computed to obtain velocity expressions, which were subsequently used to determine the kinetic energy of each system component. The total kinetic energy of the system was obtained by summing these individual contributions. The potential energy of the system elements was also formulated, incorporating restoring forces.

Applying Hamilton's Principle to the derived equations resulted in the following equation of motion:

$$EIy'''' + m\ddot{y} + Ty'' = F(x,t) - (F_1 + F_2)D(x) \tag{2.1}$$

In the above equation $F(x,t)$ is an excitation force at a single point and can be expressed as

$$F(t) = f_0 sin(\omega_e t) \tag{2.2}$$

where f_0 is the amplitude of the force, and ω_e is the excitation frequency. Further, F1, F2 and D(x) are expressed as

$$F_1 = m_d \ddot{y} \tag{2.3}$$

$$F_2 = k(y - y_d) + c(\dot{y} - \dot{y_d}) \tag{2.4}$$

$$D(x) = \delta(x - x_r) \tag{2.5}$$

The governing equation of motion for the suspended mass is given by

$$m_d \ddot{y}_d - F_2 = 0\,. \tag{2.6}$$

While a simplified mathematical model was employed to support simulation efforts and validate aspects of the experimental framework, it was not the primary focus of this study. This research emphasizes data-driven techniques for vibration monitoring and state reconstruction, rather than detailed physical modeling of damper dynamics. Unlike the work of Kakou et al [33]., which analyzed the motion of a traveling Mobile Damping Robot (MDR) through governing equations, this thesis considers only the vertical displacement response of a stationary damper. The dynamic equations related to the MDR's movement were deliberately excluded, as the primary objective here is to develop a robust sensing and predictive framework that can support future integration with mobile or adaptive damping systems.

2.2 Deep Learning Based Modeling

This section details the methodology employed for modeling power lines. A neural network model is used to predict vibration data at multiple points based on time-series inputs, enabling real-time tracking of the dynamic state of the powerline. Since only limited sensor data is available, a curve-fitting algorithm is applied to interpolate vibration patterns across the entire span of the power line. The combined approach provides a computationally efficient alternative to traditional numerical methods.

2.2.1 Data Preparation

To get vibration data from the power line, sensors are assumed to be placed at 5 points on the span of the power line as shown in 2.2.

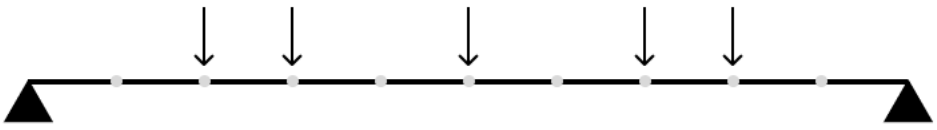


Figure 2.2: Sensor positions on the span of the powerline

To turn this problem into a supervised learning problem, we introduce *lag* into the series, [14]. *Lag* can be defined as the number of previous timesteps that is used to predict the next one.

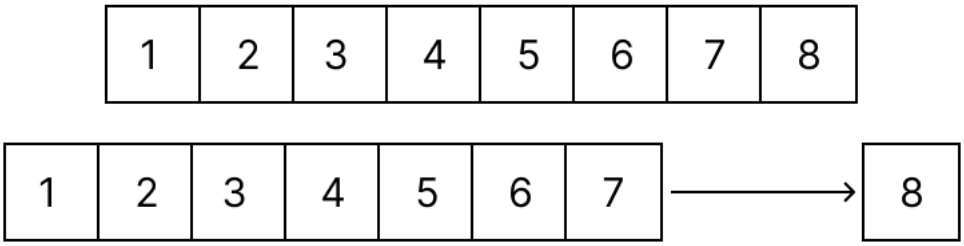


Figure 2.3: Introducing lag into the series (example)

2.2.2 Neural Network

The Neural Network (NN) is modeled as a feed-forward neural network model which utilizes LSTM layers to capture the relationships between the data and its lagged counterparts, [28]. LSTM stands for Long Short-Term Memory and it is designed to capture long-term dependencies in sequential data. LSTM layers are also capable of handling input sequences of varying lengths and is a powerful tool for modeling time series data. Also LSTM based methods are shown to be superior to statistical methods like ARIMA, [50]. The structure of the NN model is shown in 2.4.

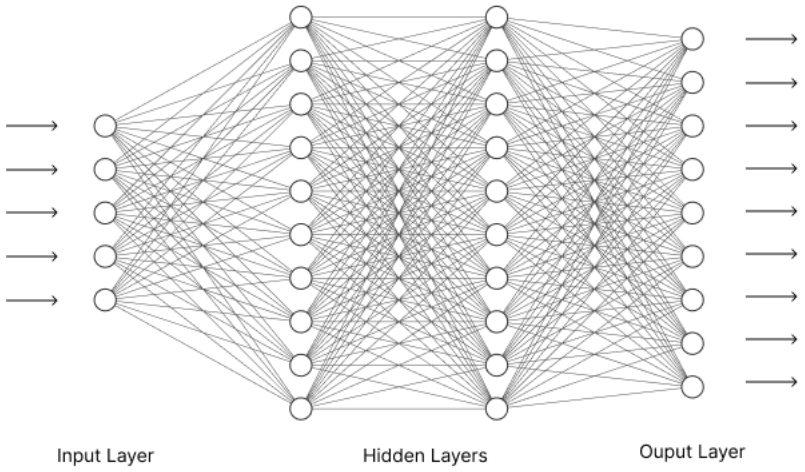


Figure 2.4: Structure of the Neural Network

The activation functions being implemented here are 'tanh' and 'linear'

$$\sigma(z) = \left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right) \qquad \sigma(z) = z \tag{2.7}$$

The model is set to train for 1000 epochs with an early stop being implemented based on the loss metric. The loss metric being used here is the mean squared error.

$$L(y, \hat{y}) = \left(\frac{1}{n}\right) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.8}$$

2.2.3 Curve Fitting

Since we only have data for 9 points on the power line, we resort to using a 'cubic spline' to interpolate values and provide us the information about the rest of the power line. Given a set of input data points (x_i, y_i) for $i = 1, 2, \ldots, n$, where x_i and y_i are the coordinates of

the i-th point, the cubic spline $S(x)$ is a piecewise-defined function that consists of cubic polynomials $S_i(x)$ defined over the intervals $[x_i, x_{i+1}]$ for $i = 1, 2, \ldots, n-1$. The polynomials are chosen such that they are continuous and have continuous first and second derivatives at the data points.

The coefficients a_i, b_i, c_i, and d_i of the polynomial $S_i(x)$ can be computed using the input data points as follows:

Let $h_i = x_{i+1} - x_i$ be the spacing between adjacent data points, and let $u_i = (y_{i+1} - y_i)/h_i$ be the slope between adjacent data points.

Then, for $i = 1, 2, \ldots, n-1$, the coefficients a_i, b_i, c_i, and d_i are given by:

$$\begin{aligned} a_i &= y_i \\ b_i &= u_i \\ c_i &= \frac{3}{h_i^2}\left(y_{i+1} - y_i\right) - \frac{1}{h_i}\left(2u_i + u_{i+1}\right) \\ d_i &= \frac{2}{h_i^3}\left(y_i - y_{i+1}\right) + \frac{1}{h_i^2}\left(u_i + u_{i+1}\right), \end{aligned} \tag{2.9}$$

where $h_i = x_{i+1} - x_i$ is the spacing between adjacent data points, and $u_i = (y_{i+1} - y_i)/h_i$ is the slope between adjacent data points.

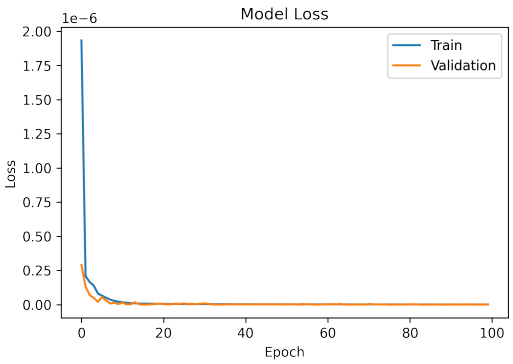
The cubic spline $S(x)$ over the interval $[x_i, x_{i+1}]$ can be evaluated for any x in that interval using the formula:

$$S_i\left(x\right) = a_i + b_i\left(x - x_i\right) + c_i\left(x - x_i\right)^2 + d_i\left(x - x_i\right)^3 \tag{2.10}$$

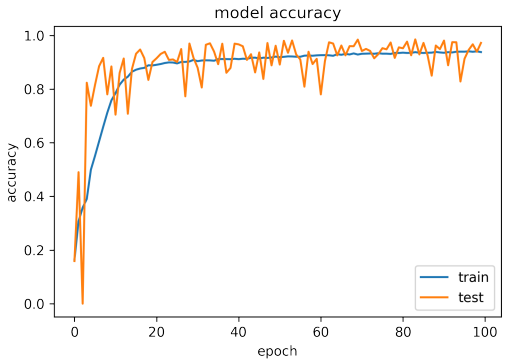
The resulting spline is smooth and continuous and passes through all the input data points (x_i, y_i).

2.2.4 Results and Discussion

The NN shows good performance. It can accurately predict the vibration information when fed with a test signal. The losses of the neural network reduce rapidly over the epochs and training stops when the losses start increasing consistently, thus minimizing over-fitting (shown in 2.5. 2.6 shows the predicted signal along with the test signal. The proposed model is able to accurately predict the dynamics of the power line and predict the vibration of 9 points from the information from 5 points.



(a) Model Loss vs Training Epoch



(b) Model Accuracy vs Training Epoch

Figure 2.5: Training Metrics of the Neural Network

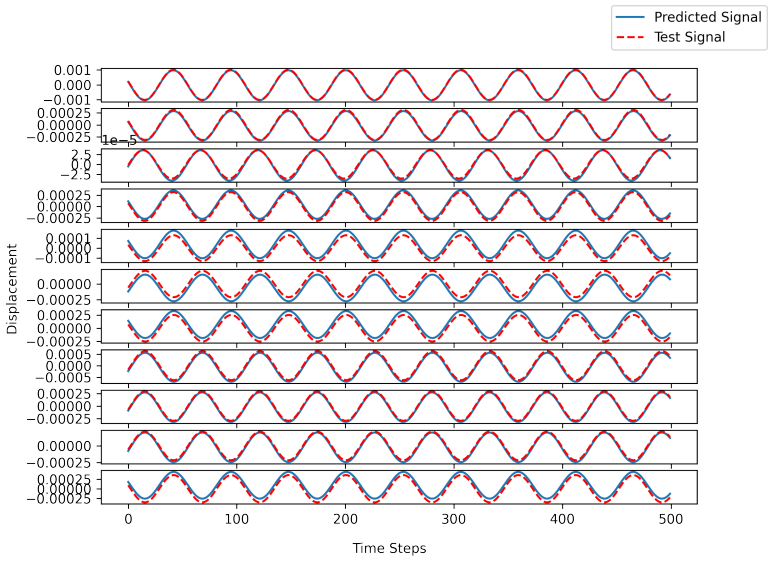


Figure 2.6: Predicted Signal vs Test Signal

The curve fitting performs as intended. The vibration data from the Matlab model and the output of the curve fitting are shown in 2.7. The blue box on the curve fitting output shows the position of the antinode. Hence, this figure confirms that the proposed data-driven model can accurately track the antinode of the vibration loop by recreating the dynamic profile of the powerline.

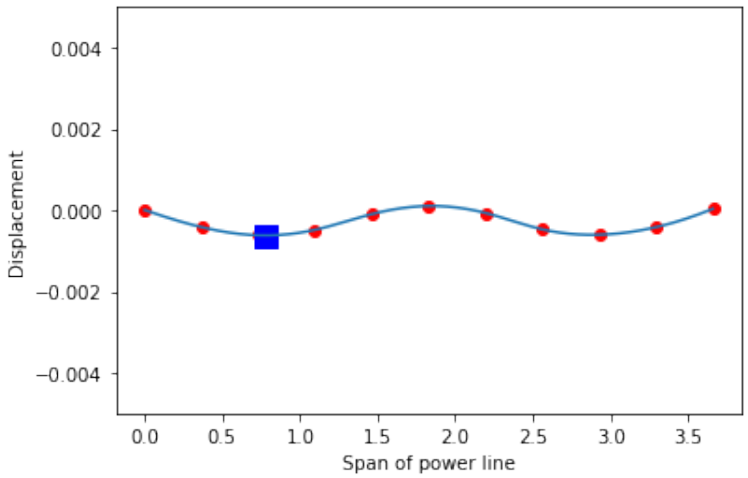


Figure 2.7: Data-driven model vs. MATLAB model

2.3 Dynamic Mode Decomposition

The application of Dynamic Mode Decomposition (DMD) in this study is pivotal for modeling complex dynamical systems based on time-series data. Unlike conventional numerical methods, which often require substantial computational resources and detailed physical modeling, DMD offers an efficient, data-centric approach to capture the underlying dynamics of the system. The essence of DMD lies in decomposing the data into spatial modes coupled with their corresponding temporal dynamics. These modes provide insights into the behavior of the system, making DMD particularly suitable for real-time applications such as dynamic state tracking of overhead powerlines.

DMD was first introduced by Peter Schmid in 2008 as a technique for analyzing fluid dynamics data [35, 47, 49]. It originated as an extension of the Proper Orthogonal Decomposition (POD) but with the advantage of incorporating temporal evolution directly into the decomposition. Since its inception, DMD has been widely adopted in various fields, including fluid mechanics, neuroscience, and structural health monitoring, due to its ability to extract dynamic patterns from high-dimensional datasets without requiring explicit knowledge of the governing equations.

Dynamic Mode Decomposition operates by identifying coherent structures within the system that evolve linearly in time, which simplifies the analysis of complex nonlinear systems. This capability is particularly advantageous when dealing with systems, such as vibrating powerlines, where direct modeling of the underlying physics is often infeasible due to the intricate interactions of various parameters. By relying on time-series data, DMD bypasses the need for explicit governing equations, thus enabling real-time implementation in scenarios where computational efficiency is paramount.

2.3.1 Time Dependent State Space

The governing discrete-time model employed in this study assumes that the state vector at the next time step, , is related to the current state vector, , via a state transition matrix, :

$$X_{k+1}^m = AX_k^m \tag{2.11}$$

Here, is constructed from time-series data, with each column representing a snapshot of the system's state at a specific time.

$$X_k = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & x_3 \cdots & x_m \\ | & | & & | \end{bmatrix} \tag{2.12}$$

and

$$X_{k+1} = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & x_4 \cdots & x_{m+1} \\ | & | & & | \end{bmatrix} \tag{2.13}$$

The matrix encapsulates the dynamic behavior of the system, capturing the essence of how the states evolve over time. To approximate , the Moore-Penrose pseudo-inverse is employed due to its ability to handle non-square and singular matrices:

This formulation ensures that the model remains robust even when the system's state space is large, the data is noisy, or the underlying dynamics exhibit stochastic behavior. The pseudo-inverse approach further enables the accommodation of high-dimensional datasets by effectively reducing the complexity of the system while preserving its critical dynamic features.

The ability of DMD to capture both spatial and temporal information is exemplified in the representation of the state transition matrix, . The eigenvalues of represent the growth or decay rates and oscillatory behavior of the modes, while the eigenvectors provide the corresponding spatial profiles. This dual characterization of the system dynamics makes DMD an invaluable tool for understanding the fundamental behavior of vibrating powerlines and predicting their response under varying conditions.

2.3.2 State Transition Matrix

The matrix A is the operator that maps X_k to X_{k+1} which essentially makes it a state-transition matrix like in state space models. Since the state vectors are not invertible, we resort to the Moore-Penrose pseudo-inverse. The Moore-Penrose pseudo-inverse is a generalization of the matrix inverse for non-square and singular matrices. Given a matrix A, its pseudo-inverse denoted as A^+ is unique and satisfies the following properties:

$$\begin{aligned} A\left(A^+\right)A &= A \\ \left(A^+\right)A\left(A^+\right) &= A^+ \\ \left(A\left(A^+\right)\right)^* &= A\left(A^+\right) \\ \left(\left(A^+\right)A\right)^* &= \left(A^+\right)A \end{aligned} \tag{2.14}$$

, where $*$ represents the conjugate transpose operation. When a linear system $Ax - b$, has no exact solution, the pseudo-inverse can be used to find the least squares solution. The vector $x = A^+b$ minimizes the Euclidean norm of the residual $|Ax - b|$.

So, the state-transition matrix can be approximated as:

$$A \approx X_{k+1}X_k^+ \tag{2.15}$$

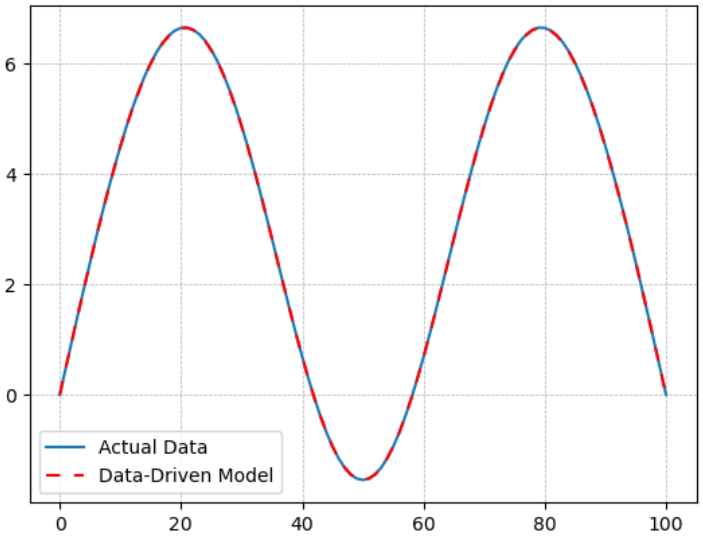


Figure 2.8: State Transition Matrix

As seen in 2.8, it is evident that A is a good approximation of the state-transition matrix.

2.4 Stability of the system

To analyze the stability of the system, the eigenvalues of A are studied. The eigenvalues denoted as λ_i give insight into the temporal behavior and stability of the system. In a discrete-time system, the system is stable if all the eigenvalues fall within the unit circle in the complex plane i.e.:

$$|\lambda_i| < 1 \forall i \in 1, 2, \ldots, r \tag{2.16}$$

where r is the number of modes. If any eigenvalue has a magnitude greater than or equal to 1, the corresponding mode will grow exponentially over time, indicating unstable behavior. Eigenvalues with magnitudes of less than 1 correspond to stable modes that decay over time.

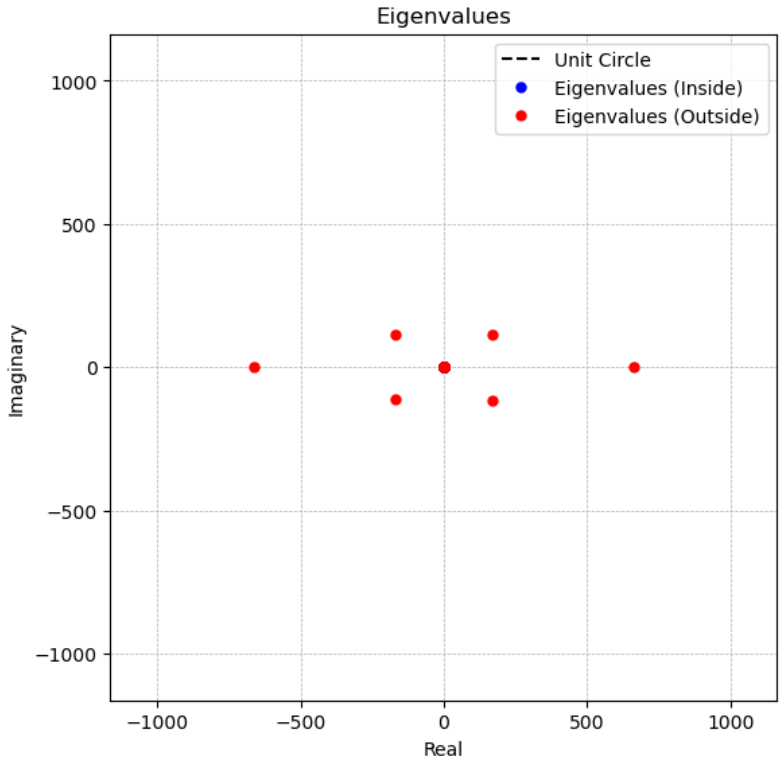


Figure 2.9: Eigenvalues before Stabilization

In 2.9, the maximum values of the imaginary and real parts are 114.98 and 663.46. Out of 101 eigenvalues, 11 are outside the unit circle and these eigenvalues correspond to modes that will grow exponentially. This means that our system behavior will also grow exponentially and hence it is unstable. So, the system has to be stabilized while making sure that A is still a good approximation of the state-transition matrix.

2.5 Tikhonov Regularization

To stabilize the system, Tikhonov Regularization was integrated into the DMD framework. This method modifies the optimization problem by adding a penalty term to the objective function, effectively discouraging large values in and thereby promoting stability [27, 53].

The regularized objective function is expressed as:

$$\min_{A} |X_{k+1} - AX_k|_F^2 + \alpha |A|_F^2 \tag{2.17}$$

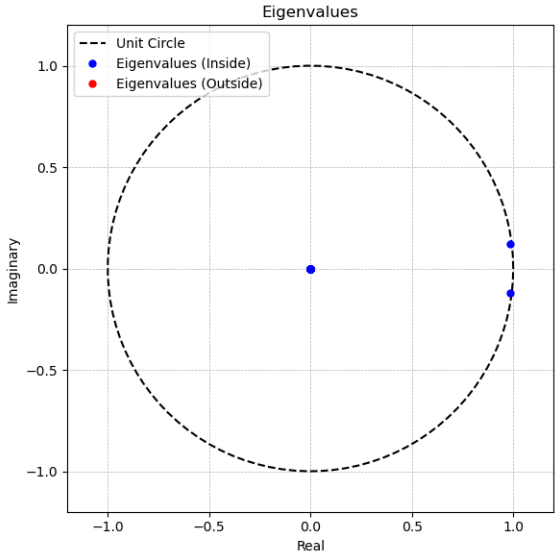
where $\alpha > 0$ is the regularization parameter, $|\cdot|_F$ denotes the Frobenius norm. The regularization term $\alpha|A|_F^2$ penalizes large values in the matrix A, promoting stability. Then the regularized state-transition matrix, A_{reg} can be calculated as:

$$A_{reg} = X_{k+1}X_k^T(X_kX_k^T + \alpha I)^{-1} \tag{2.18}$$

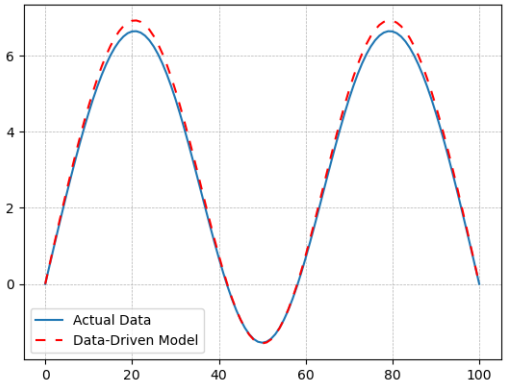
where I is the identity matrix.

The choice of α is critical; while a larger value improves stability, excessive regularization may lead to an overly simplistic model that fails to capture key dynamic features. Thus, an optimal value of must be selected through careful analysis and experimentation.

Regularization not only stabilizes the system but also enhances its resilience to noise and data imperfections. By penalizing large variations, Tikhonov Regularization ensures that the resulting model remains consistent and reliable, even under adverse conditions. This robustness is particularly important for applications involving real-time state estimation, where measurement errors and environmental fluctuations are inevitable.



(a) Eigenvalues after Regularization



(b) State Transition Matrix after Regularization

Figure 2.10: Model after Regularization

In 2.10a, the maximum values of the imaginary and real parts are 0.1212 and 0.9868 respectively. All the eigenvalues lie within the unit circle and hence the system is stable. As seen in fig. 2.10b, A_{reg} is still a good approximation of the state-transition matrix. The choice of the regularization parameter, α is crucial for achieving the perfect balance in A_{reg}. A larger value of α will push the eigenvalues to the origin thus improving stability, but if α is too large the resulting A_{reg} may not be able to accurately capture the dynamics of the system.

Chapter 3

Real Time Estimation

Real-time estimation of dynamic system states is critical for effective monitoring and control of wind-induced vibrations in overhead power lines. Given the practical limitations on sensor coverage and the presence of noise and data loss, robust estimation techniques are needed to infer the full vibration profile from limited observations. This chapter presents a suite of real-time estimation methods—including optimal sensor placement, Kalman filtering, particle filtering, and compressive sensing—that work in conjunction with the data-driven models developed earlier. Together, these tools enable accurate reconstruction of the powerline's dynamic state using sparse and potentially noisy sensor data, laying the groundwork for adaptive vibration mitigation and intelligent infrastructure monitoring.

3.1 Sensor Placement

3.1.1 SVD/POD and Pivoted QR

In order to reconstruct the vibration profile of a beam from a limited number of measurement points, one can exploit a low-dimensional representation learned from the full displacement data [15]. Suppose the beam is discretized into N points along its length, and that we observe M snapshots in time. Let $\mathbf{W} \in \mathbb{R}^{N\times M}$ be the matrix whose i-th row collects the time-history of the i-th spatial point's displacement. The key steps are as follows:

1. **Singular Value Decomposition (SVD).**

   Compute the SVD of $\mathbf{W}$:

   $$\mathbf{W} \;=\; \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\mathsf{T}},$$

   where $\mathbf{U} \in \mathbb{R}^{N\times N}$ and $\mathbf{V} \in \mathbb{R}^{M\times M}$ have orthonormal columns, and $\mathbf{\Sigma}$ is diagonal (containing the singular values in non-increasing order). The columns of $\mathbf{U}$ are often referred to as the *POD modes* or *principal spatial components* [41].

2. **Mode Truncation.**

   Observe that the singular values $\sigma_1 \geq \sigma_2 \geq \cdots$ typically decay, indicating that a low-rank approximation may capture most of the energy in $\mathbf{W}$. Select r so that

   $$\sum_{k=1}^{r} \sigma_k^2 \Big/ \sum_{k=1}^{\min(N,M)} \sigma_k^2 \;\approx\; \alpha \quad \text{(e.g. 90\% or 95\% of the total energy)},$$

   and define

   $$\mathbf{U}_r \;=\; \mathbf{U}(:,\, 1{:}r).$$

   This matrix $\mathbf{U}_r \in \mathbb{R}^{N\times r}$ spans the dominant r-dimensional subspace of the data.

3. **Pivoted QR for Sensor Selection.**

   We wish to measure *only a subset* of the N beam points while still being able to recover (approximately) the entire displacement $\mathbf{w}(t)$ at any time. Equivalently, we want to select a subset of rows of $\mathbf{U}_r$ that maintains the ability to solve for the r modal coefficients [38].

   Concretely, let $\mathbf{U}_r^{\mathsf{T}} \in \mathbb{R}^{r\times N}$. A *pivoted QR decomposition* factorizes

   $$\mathbf{U}_r^{\mathsf{T}}\,\mathbf{\Pi} \;=\; \mathbf{Q}\,\mathbf{R},$$

where $\mathbf{\Pi}$ is a permutation matrix that greedily chooses the "best" columns. The first r pivot columns (in the permuted order) identify the row indices of $\mathbf{U}_r$ that best span its range. Denote these row indices by

$$\mathcal{S}(r) \;=\; \{\, j_1,\, j_2, \ldots, j_r\}.$$

Placing sensors at those beam positions (indexed by $\mathcal{S}(r)$) ensures that the submatrix $\mathbf{U}_r(\mathcal{S}(r), :)$ is of full rank and thus invertible (or well-conditioned in a least-squares sense).

4. **Reconstruction.**

   Once $\mathcal{S}(r)$ is chosen, any new measurement $\mathbf{y}(t) \in \mathbb{R}^r$ (the displacements at the selected sensors) yields the modal coefficients $\boldsymbol{\alpha}(t)$ via ([18])

$$\mathbf{y}(t) \;=\; \mathbf{U}_r\big(\mathcal{S}(r), :\big)\, \boldsymbol{\alpha}(t).$$

   Solving for $\boldsymbol{\alpha}(t)$, we then reconstruct the *entire* displacement $\hat{\mathbf{w}}(t) \in \mathbb{R}^N$ by

$$\hat{\mathbf{w}}(t) \;=\; \mathbf{U}_r\, \boldsymbol{\alpha}(t).$$

The principal advantage of this approach is that *one set of sensors* can reliably recover all snapshots in the data matrix (and similar future states, assuming the same modal structure). By incrementing r from 1 up to a chosen maximum, one obtains different $\mathcal{S}(r)$ sets for each truncation level, providing a direct way to study how many modes (and consequently how many sensors) are needed to achieve a given accuracy. 3.1 shows the optimal sensor positions for different mode truncations.

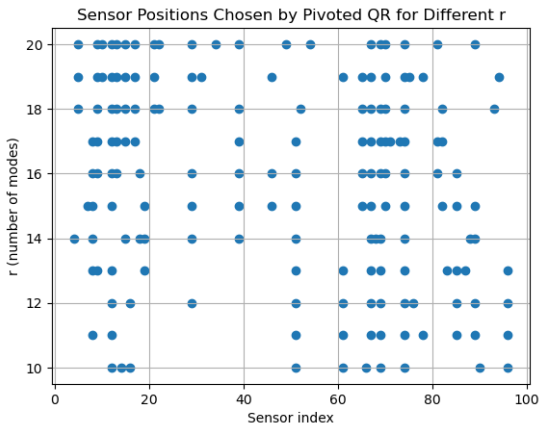


Figure 3.1: Optimal Sensor positions for different Mode Truncations

3.2 Particle Filter

In this section, we describe the implementation of a particle filter to estimate missing sensor data in a vibrating beam system modeled using Dynamic Mode Decomposition (DMD). Given a learned state transition model of the form

$$X_{k+1} = AX_k, \tag{3.1}$$

where $X_k \in \mathbb{R}^n$ represents the state of the system at time step k, and $A \in \mathbb{R}^{n\times n}$ is the transition matrix derived via DMD, a particle filter is employed to recursively estimate unobserved sensor values [5].

The system is formulated as a state-space model:

$$X_{k+1} = AX_k + w_k, \quad w_k \sim \mathcal{N}(0, Q), \tag{3.2}$$

$$Y_k = HX_k + v_k, \quad v_k \sim \mathcal{N}(0, R), \tag{3.3}$$

where:

- $X_k \in \mathbb{R}^n$ is the full system state (sensor measurements),
- $Y_k \in \mathbb{R}^m$ is the subset of available sensor measurements ($m < n$),
- $H \in \mathbb{R}^{m \times n}$ is a measurement matrix selecting the available sensor readings,
- $w_k \in \mathbb{R}^n$ is the process noise with covariance Q, modeling unmodeled dynamics,
- $v_k \in \mathbb{R}^m$ is the measurement noise with covariance R.

A particle filter is a Monte Carlo-based Bayesian estimation technique that approximates the posterior distribution $p(X_k|Y_{1:k})$ using a set of weighted particles [21]. The filter starts with an initial set of N particles sampled from a prior distribution:

$$X_k^{(i)} \sim p(X_0), \quad w_k^{(i)} = \frac{1}{N}, \quad i = 1, \ldots, N. \tag{3.4}$$

Prediction Step

Each particle is propagated through the process model:

$$X_{k+1}^{(i)} = AX_k^{(i)} + w_k^{(i)}, \quad w_k^{(i)} \sim \mathcal{N}(0, Q). \tag{3.5}$$

Update Step

When sensor measurements Y_k are available, particle weights are updated based on the likelihood of the observed data:

$$w_{k+1}^{(i)} \propto p(Y_{k+1}|X_{k+1}^{(i)}) w_k^{(i)}, \tag{3.6}$$

where the likelihood is assumed Gaussian:

$$p(Y_{k+1}|X_{k+1}^{(i)}) \propto \exp\left(-\frac{1}{2}(Y_{k+1} - HX_{k+1}^{(i)})^T R^{-1}(Y_{k+1} - HX_{k+1}^{(i)})\right). \tag{3.7}$$

After computing the weights, normalization is performed:

$$w_{k+1}^{(i)} \leftarrow \frac{w_{k+1}^{(i)}}{\sum_{j=1}^{N} w_{k+1}^{(j)}}. \tag{3.8}$$

Resampling Step

To prevent particle degeneracy, resampling is performed by selecting new particles according to their weights:

$$X_{k+1}^{(i)} \sim \sum_{j=1}^{N} w_{k+1}^{(j)} \delta(X_{k+1} - X_{k+1}^{(j)}), \tag{3.9}$$

where $\delta(\cdot)$ is the Dirac delta function. The weights are then reset:

$$w_{k+1}^{(i)} = \frac{1}{N}, \quad \forall i. \tag{3.10}$$

Final State Estimation

The estimated state at time $k+1$ is obtained as the weighted mean:

$$\hat{X}_{k+1} = \sum_{i=1}^{N} w_{k+1}^{(i)} X_{k+1}^{(i)}. \tag{3.11}$$

This approach enables the estimation of unmeasured sensor values in real-time by leveraging the learned DMD dynamics while accounting for measurement uncertainty through the particle filtering framework [52].

3.3 Kalman Filter

The Kalman Filter operates on the principle of prediction and correction, following a two-step process that first predicts the system's future states and then corrects the prediction based on new measurement data. This allows the filter to estimate the states in real time and predict all states of the system even when the system is only partially observable.

A tailored Kalman Filter is implemented to work with the data-driven workflow and the DMD-based state transition model [51]:

$$X_{k+1} = AX_k + w_k, \tag{3.12}$$

where $X_k \in \mathbb{R}^n$ represents the system state at time step k, and $A \in \mathbb{R}^{n\times n}$ is the transition matrix derived via DMD.

Prediction Step

The predicted state estimate and its covariance are computed as follows:

$$\hat{X}_{k|k-1} = A\hat{X}_{k-1|k-1} + Q, \tag{3.13}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q + \epsilon I, \tag{3.14}$$

where:

- $\hat{X}$ is the estimated state vector,
- Q is the process noise covariance,
- P is the error covariance matrix,

- ϵ is a small regularization term to ensure numerical stability.

Update Step

Upon receiving new measurement data, the filter updates its state estimate:

$$Y_k = Z_k - H_k \hat{X}_{k|k-1}, \tag{3.15}$$

$$S_k = H_k P_{k|k-1} H_k^T + R + \epsilon I, \tag{3.16}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1}, \tag{3.17}$$

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k Y_k, \tag{3.18}$$

$$P_{k|k} = (I - K_k H) P_{k|k-1}. \tag{3.19}$$

where:

- Z_k is the new measurement data,
- Y_k is the measurement residual (innovation),
- S_k is the innovation covariance,
- K_k is the Kalman Gain,
- R is the measurement noise covariance,
- ϵ is a small value ensuring numerical stability.

The regularization term ϵ prevents the innovation covariance from approaching singularity [39]. Additionally, this implementation assumes that only a subset of states in the system are observable, and the filter updates estimates accordingly.

Final State Estimation

The estimated state at time k is obtained as:

$$\hat{X}_k = \hat{X}_{k|k}. \tag{3.20}$$

This Kalman Filter effectively utilizes the learned DMD transition model while incorporating measurement updates to estimate missing sensor values in real-time.

3.4 Compressive Sensing

Sensors are crucial in collecting and transmitting data. However, sensor networks are often susceptible to data loss due to factors such as transmission errors, power constraints, and environmental interference [6]. Missing data can significantly impact the reliability and accuracy of downstream analyses, making effective reconstruction techniques essential.

Compressive sensing (CS) has emerged as a powerful approach for reconstructing incomplete signals by leveraging the inherent sparsity of many real-world signals. Unlike traditional signal recovery methods that require a high sampling rate, CS enables accurate reconstruction from a limited number of measurements, making it particularly well-suited for scenarios with constrained resources and sporadic data loss. By exploiting the sparsity of signals in a suitable transform domain, CS allows for robust and efficient recovery, offering a promising solution to mitigate the challenges posed by missing sensor data.

This section explores the problem of data loss in sensor networks and how compressive sensing can be utilized to reconstruct lost information, ensuring reliable and high-quality signal recovery.

3.4.1 Mathematical Formulation

CS is based on two key principles [17]:

- **Sparsity:** Many real-world signals have a sparse representation in a specific basis, meaning they can be expressed using only a few significant coefficients.

- **Incoherence:** The measurement process should be sufficiently uncorrelated with the sparsity basis to ensure effective reconstruction.

A signal $x \in \mathbb{R}^N$ is **sparse** in a certain basis Ψ, meaning it can be represented as [16]:

$$x = \Psi s \tag{3.21}$$

where Ψ is an orthonormal basis (e.g., Fourier, Wavelet, or Discrete Cosine Transform), and s is a sparse coefficient vector.

Instead of acquiring all N samples, we take only M ($M \ll N$) measurements using a sensing matrix Φ, obtaining:

$$y = \Phi x \tag{3.22}$$

Substituting $x = \Psi s$:

$$y = \Phi \Psi s = \Theta s \tag{3.23}$$

where $\Theta = \Phi\Psi$ is the **compressed sensing matrix**. Since $M < N$, this system is *under-determined*, meaning additional constraints are required for signal recovery.

3.4.2 Basis Pursuit for Signal Reconstruction

To recover a sparse signal, Basis Pursuit (BP) seeks the sparsest solution satisfying the measurement equation [20]. This is achieved by solving:

$$\hat{s} = \arg\min_{s} \|s\|_1 \quad \text{subject to} \quad y = \Theta s \tag{3.24}$$

where the ℓ_1**-norm** is defined as:

$$\|s\|_1 = \sum_i |s_i| \tag{3.25}$$

Minimizing the ℓ_1-norm promotes sparsity, making this approach effective for reconstructing signals with few nonzero coefficients.

In the presence of noise:

$$y = \Theta s + e \tag{3.26}$$

where e represents noise, the problem is relaxed to:

$$\hat{s} = \arg\min_{s} \|s\|_1 \quad \text{subject to} \quad \|y - \Theta s\|_2 \leq \epsilon \tag{3.27}$$

where $\|y - \Theta s\|_2$ ensures consistency with observed data.

Several methods exist to solve Basis Pursuit problems, including:

- **Linear Programming (LP)**: Reformulates ℓ_1-minimization as a linear program.
- **Iterative Shrinkage-Thresholding Algorithm (ISTA)**: A gradient-based approach with soft-thresholding.
- **Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)**: Accelerates ISTA using momentum updates.

- **Alternating Direction Method of Multipliers (ADMM)**: A flexible and efficient optimization method.

This implementation of Basis Pursuit reconstructs a signal by leveraging the **Discrete Cosine Transform (DCT)** domain, where the signal is sparse.

Given observations y, the reconstruction problem is:

$$\hat{x} = \arg\min_{x} \|x\|_1 \quad \text{subject to} \quad Ax = y \tag{3.28}$$

where A is the undersampled IDCT matrix.

The advantages of this method are:

- **Efficient Recovery**: Requires fewer samples than traditional methods.
- **Robustness to Noise**: BP is resilient to measurement noise.
- **Computational Feasibility**: Modern solvers efficiently handle large problems.

To demonstrate this method, a point is randomly considered on the span on the powerline. Here, the midpoint of the powerline is considered. 3.2 shows the signal received from the sensor at point 50 i.e., the midpoint. The time chosen is the first 5000 time steps.

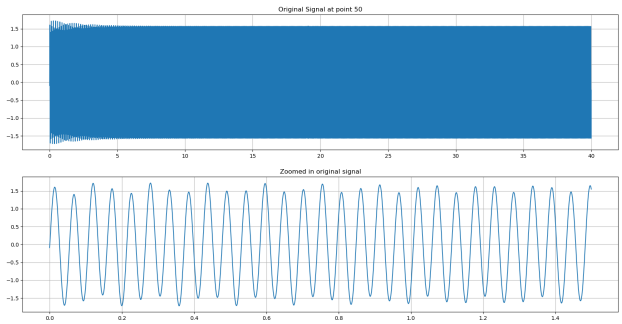


Figure 3.2: Original Signal and Zoomed in Signal

Out of 5000 timesteps, 300 time steps are chosen to be sampled at random. 3.3 shows the sampling and the signal reconstructed using only the sampled points.

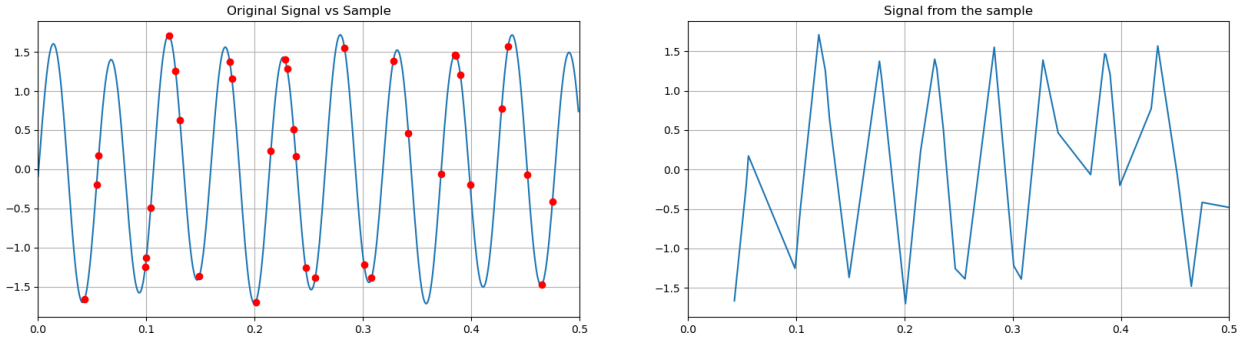


Figure 3.3: Sample and Reonstructed Signal

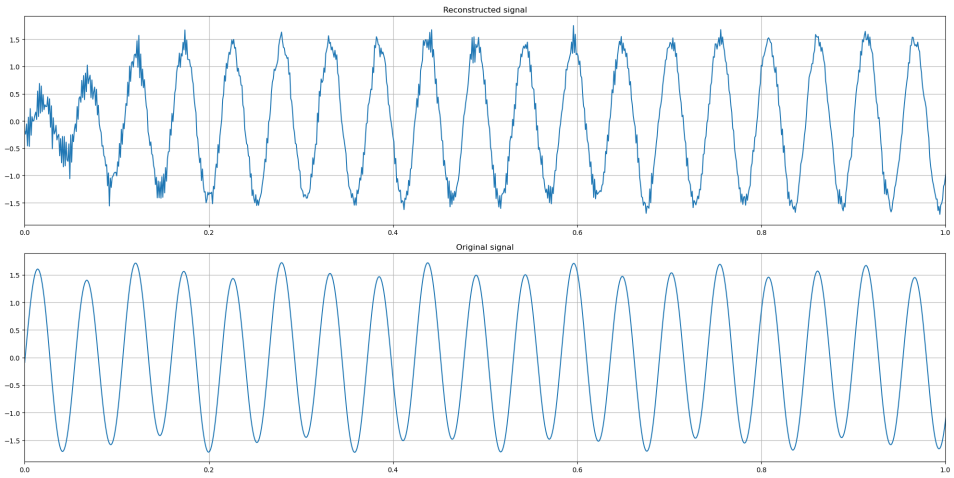


Figure 3.4: Reconstructed Signal vs Original Signal using Compressive Sensing

3.4 shows the reconstructed signal along with the original signal. This shows that the use of Compressive Sensing is a viable method to deal with data loss from sensors.

3.4.3 Compressing Sensing on Span of Powerline

The method outlined above can also be implemented over the span of the powerline, considering the sensor placement points as points to be sampled, the span of the powerline as

the duration of the signal to reconstruct the dynamic state of the powerline. 3.5 shows this implementation.

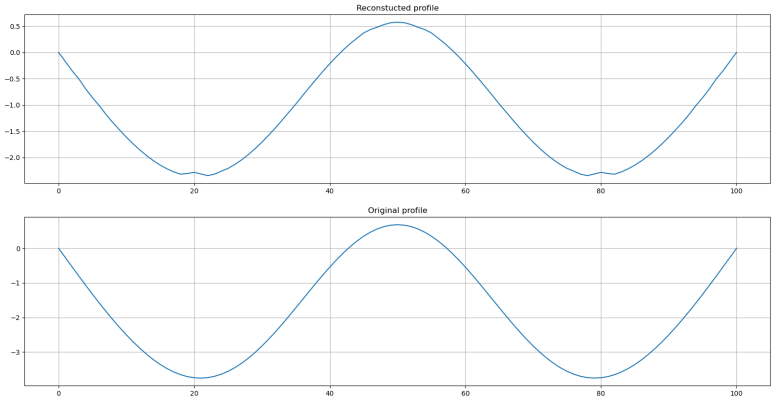


Figure 3.5: Reconstructed Profile vs Original Profile

This method can be used in conjunction with other methods to accurately reconstruct the dynamic state of the powerline at every time step. This has not been explored further in this thesis but can be a very promising area for further research.

Chapter 4

Results

This chapter presents the results obtained from the implementation of the modeling, estimation, and reconstruction techniques discussed in the previous chapters. The performance of the LSTM-based neural network, Dynamic Mode Decomposition (DMD), Kalman and particle filters, and compressive sensing is evaluated in terms of accuracy, stability, and robustness. Particular attention is given to the effectiveness of sensor placement strategies and the ability to reconstruct the full dynamic profile of the powerline from limited measurements. These results validate the proposed framework's ability to track and predict wind-induced vibrations in real time, demonstrating its potential for practical deployment in transmission line monitoring systems.

4.1 Dynamic Profile Reconstruction

4.1.1 SVD/POD with Pivoted QR

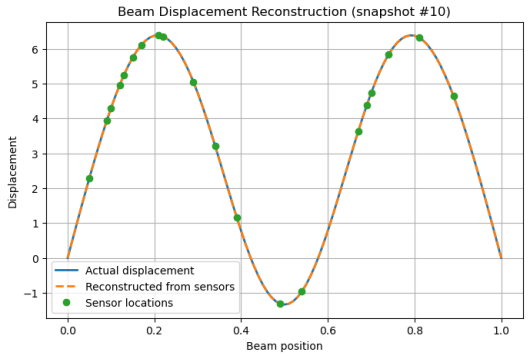
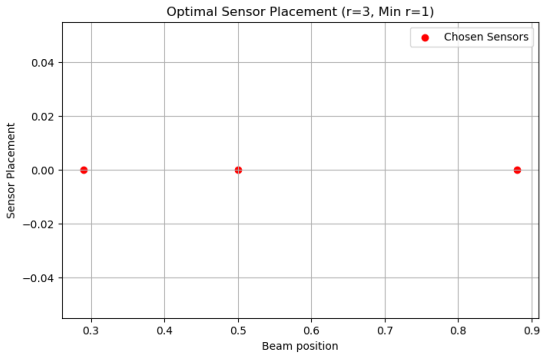
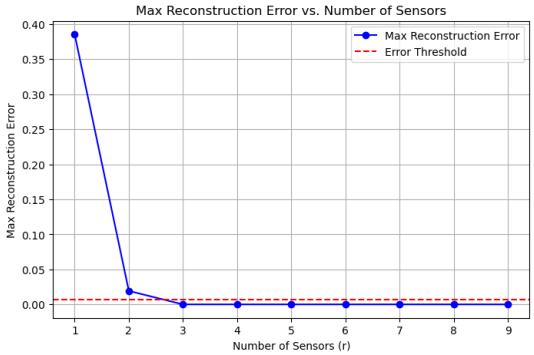


Figure 4.1: Reconstructed Profile of the Powerline (r=20)



(a) Optimal Sensor Placement



(b) Reconstruction Error vs Number of Sensors

Figure 4.2: For Standalone Powerline

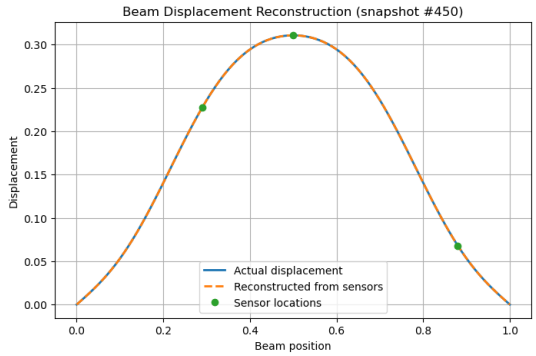
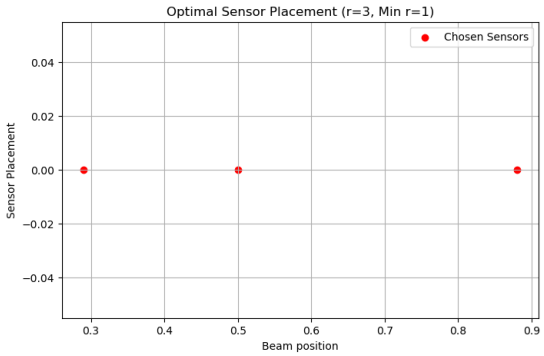
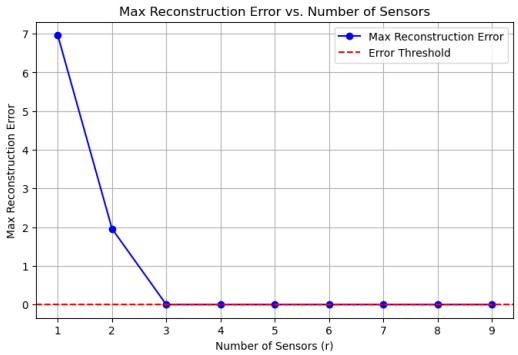


Figure 4.3: Reconstructed Profile of the Powerline



(a) Optimal Sensor Placement (with mass)



(b) Reconstruction Error vs Number of Sensors (with Mass)

Figure 4.4: For Powerline with mass at midspan

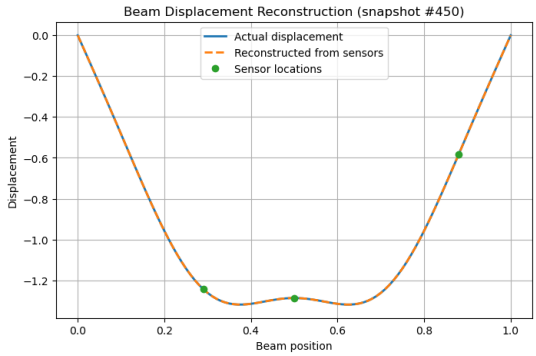
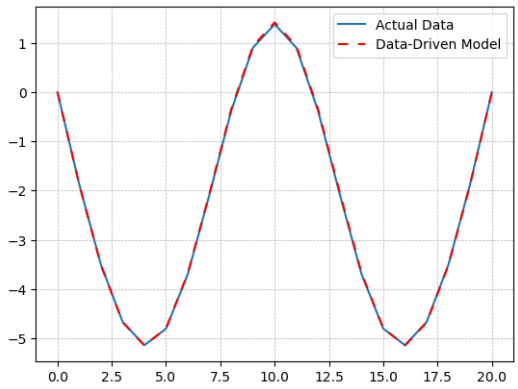


Figure 4.5: Reconstructed Profile of the Powerline (with mass)

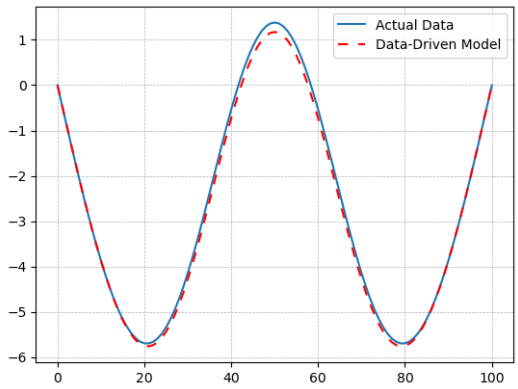
The results presented in 4.1 through 4.5 demonstrate the effectiveness of using SVD/POD in conjunction with pivoted QR decomposition for optimal sensor placement and dynamic profile reconstruction. The reconstructed vibration profiles align closely with the ground truth, indicating that the dominant mode shapes of the system are well captured by a reduced number of sensors. As shown in the reconstruction error plots, increasing the number of sensors (i.e., the number of retained modes ie, r) leads to a rapid reduction in estimation error, confirming that most of the system's dynamic energy is contained within a few dominant modes. The sensor placement determined via pivoted QR ensures that the selected points span the reduced subspace effectively, enabling accurate reconstruction without the need for exhaustive spatial coverage. In the standalone powerline case, even a modest number of sensors is sufficient to capture the full displacement profile with minimal deviation. When a mass is introduced at midspan, the algorithm still identifies sensor locations that preserve the ability to reconstruct altered mode shapes, suggesting that the method is robust to structural perturbations. This approach not only reduces hardware requirements but also enhances the observability of the system by ensuring that the selected sensor locations provide maximal information about the full-state dynamics. These results support the feasibility of deploying sparse sensor networks for large-scale, distributed vibration monitoring, particularly in applications where cost, weight, or accessibility constraints limit the number

of available sensors.

4.1.2 DMD with Kalman Filter



(a) Span divided into 21 points



(b) Span divided into 101 points

Figure 4.6: Reconstructed Profile

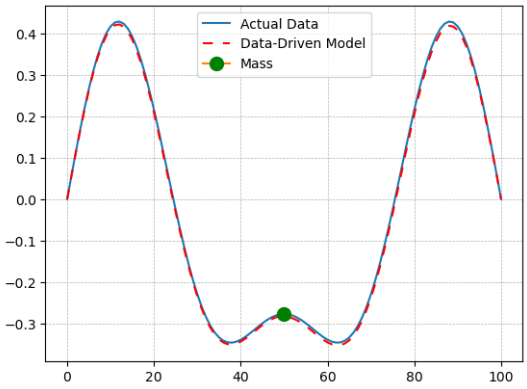


Figure 4.7: Reconstructed profile (with mass)

The reconstructed profiles shown in 4.6 and 4.7 highlight the capability of the DMD with Kalman Filter framework to accurately estimate the dynamic state of the powerline using only a limited number of sensor inputs. In the standalone powerline case (4.6), the reconstructed vibration profile closely aligns with the true profile across both coarse and fine spatial resolutions (21 and 101 discretization points, respectively). This demonstrates that the approach is robust to changes in spatial granularity and can maintain estimation accuracy across varying levels of system detail.

In the scenario with an attached mass at midspan (4.7), the method successfully adapts to the modified vibration characteristics introduced by the added inertia. Despite the local perturbation to the system dynamics, the reconstructed profile remains consistent with the actual signal. This shows that the estimator is able to generalize to structural changes in the conductor, which is essential for real-world deployments where such irregularities may be present due to environmental loading or hardware installations.

The integration of DMD and Kalman filtering offers a practical balance between model expressiveness and real-time performance. The reduced-order DMD model captures the dominant system behavior, while the Kalman filter continuously corrects the estimated state based on sparse measurements. This combination enhances resilience to noise and measurement uncertainty, enabling the reconstruction of full-state vibration profiles in scenarios where only partial observability is available.

Overall, these results validate the use of DMD with Kalman Filtering as a viable and efficient method for tracking conductor vibrations in real time. Its ability to maintain performance under sparse sensing and system variability makes it well-suited for integration with future autonomous inspection or damping platforms.

Chapter 5

Discussion

The results presented in this thesis highlight the advantages of integrating data-driven methods into the monitoring and mitigation of wind-induced vibrations (WIV) in power transmission systems. Traditional approaches, while grounded in well-established physical principles, are often limited in adaptability, particularly in real-time or high-variability environments. This research bridges that gap by proposing and validating a hybrid framework that combines mathematical modeling, machine learning, and real-time estimation.

5.1 Neural Network-Based Vibration Prediction

The application of LSTM-based neural networks for vibration prediction represents a significant departure from conventional, physics-based modal analysis. By training on sparse time-series data from only five sensors, the model was able to infer vibration behavior at nine points along the conductor with high fidelity. The success of this architecture lies in the LSTM's ability to preserve long-range temporal dependencies and recognize recurring dynamic patterns, which are typical of Aeolian vibrations.

The curve-fitting step, using cubic splines, complemented the neural model by interpolating vibration behavior between known points, thus enabling a continuous spatial profile of the conductor. The model's accuracy in identifying antinodes—the critical points of maximum displacement—was confirmed by comparing reconstructed profiles to simulated ground truth

from MATLAB-based models. This antinode localization is essential for effective placement of dampers or mobile vibration absorbers, particularly under changing wind conditions.

Importantly, this data-driven approach provides a scalable solution for real-world deployment, where sensor count may be limited due to cost or logistical constraints. Moreover, the modularity of the neural architecture allows for retraining and updating with new data, making the system adaptable to seasonal changes or line aging.

5.2 Dynamic Mode Decomposition and Stability Enhancement

Dynamic Mode Decomposition (DMD) was employed to extract a linear, reduced-order model of the system dynamics directly from sensor data. While DMD is powerful for decomposing complex behavior into temporal modes, its naive application can lead to unstable systems when the resulting transition matrix includes eigenvalues with magnitude greater than one.

To address this, Tikhonov Regularization was introduced to the DMD formulation, effectively stabilizing the system without sacrificing modeling accuracy. The regularized transition matrix A_{reg} retained the key modal structures while ensuring all eigenvalues lay within the unit circle. The comparative eigenvalue spectra before and after regularization clearly demonstrated this improvement. Furthermore, sensitivity analysis on the regularization parameter indicated a balance point between model fidelity and dynamic stability, suggesting that adaptive tuning of could be beneficial in future implementations.

The resulting model offers a robust foundation for real-time estimation and control tasks. By capturing the essential temporal evolution of conductor vibrations, DMD provides a middle

ground between full physical simulation and purely black-box machine learning models.

5.3 Real-Time State Estimation with Kalman and Particle Filters

The integration of state estimation algorithms, particularly Kalman and particle filters, extends the utility of the DMD model into real-time monitoring applications. The Kalman Filter, leveraging a linear-Gaussian framework, was shown to be highly effective in fusing model predictions with partial sensor measurements, offering smooth and accurate state reconstructions.

Meanwhile, the particle filter offered a non-parametric alternative capable of handling nonlinearities and non-Gaussian noise, albeit with higher computational overhead. It provided valuable robustness in scenarios with sparse or noisy data, such as sensor dropouts or bursty transmission errors. The agreement between particle filter estimates and the known ground truth validated the use of these methods in mission-critical applications.

This dual-estimator setup also allows for system redundancy. Kalman filtering can be used in nominal conditions, while the particle filter acts as a fallback or cross-validation mechanism during anomalous events.

5.4 Compressive Sensing for Data Loss Recovery

The compressive sensing (CS) framework explored in this study addressed a vital problem in structural health monitoring—sensor data loss. By exploiting the sparsity of vibration signals in the Discrete Cosine Transform (DCT) domain, the CS method successfully reconstructed

missing data from as little as 6% of the original time-series measurements.

This performance opens the door to reduced data transmission requirements, particularly in remote deployments where bandwidth is limited. Additionally, the results suggest the potential for CS-based reconstruction not only temporally (per sensor) but also spatially (across the powerline span), making it an ideal candidate for integration with UAV- or robot-based inspection platforms.

5.5 Sensor Placement via Pivoted QR Decomposition

A central component of this framework was the sensor placement strategy based on pivoted QR decomposition of the Proper Orthogonal Decomposition (POD) modes. By applying SVD to the full data matrix and then selecting the most linearly independent rows using QR with column pivoting, the method identified sensor locations that maximally preserved the modal structure of the system.

This approach provided two major benefits:

Sensor Efficiency: It allowed reconstruction of the full dynamic state using the minimum number of sensors without sacrificing accuracy.

Estimation Reliability: The QR-selected positions ensured the measurement matrix was well-conditioned, improving the robustness of downstream estimation algorithms like Kalman and particle filters.

Ultimately, this sensor placement framework helped reduce sensing overhead while enhancing the observability and stability of the vibration tracking system—critical features for large-scale or remote infrastructure deployments.

5.6 System-Level Implications

Taken together, the methods developed in this thesis constitute a holistic, data-driven pipeline for intelligent vibration monitoring and mitigation. The neural network and DMD models provide predictive capability, while Kalman and particle filters ensure robustness and observability. The addition of compressive sensing enhances fault tolerance, ensuring resilience in the face of data degradation.

Such a framework lays the groundwork for autonomous, adaptive vibration control systems in the future, potentially incorporating mobile dampers or smart inspection robots that respond dynamically to real-time feedback.

Chapter 6

Conclusions

This research explored the modeling, estimation, and reconstruction of wind-induced vibrations in overhead power transmission lines through a suite of data-driven techniques. The motivation stemmed from the critical need for scalable, real-time, and adaptive solutions to manage vibration-induced fatigue in aging grid infrastructure.

6.1 Key Findings

- LSTM-Based Neural Network Modeling: Demonstrated high predictive accuracy for vibration patterns using minimal sensor input, enabling effective antinode tracking for vibration mitigation.
- Curve Fitting with Cubic Splines: Provided a practical interpolation tool that, when coupled with neural predictions, generated full-span vibration profiles.
- DMD and Regularization: Enabled low-order system modeling directly from sensor data, with Tikhonov Regularization ensuring dynamic stability and robustness.
- Real-Time Estimation: Kalman and particle filters enabled the reconstruction of unobserved states, ensuring system observability even under partial sensor availability.
- Compressive Sensing: Proved effective for recovering missing or under-sampled data, adding resilience to sensor failures or transmission limitations.

6.2 Practical Implications

The combined approach has significant implications for transmission line asset management. Utilities could leverage such a system to:

- Optimize damper placement dynamically.
- Deploy inspection robots that respond in real-time.
- Reduce costs associated with manual inspection and unplanned maintenance.
- Integrate predictive maintenance into grid reliability planning.

Moreover, the modularity of the pipeline allows for integration with existing SCADA systems or deployment via edge computing units on smart substations.

Chapter 7

Summary

This thesis presents a comprehensive framework for the monitoring, modeling, and mitigation of wind-induced vibrations—specifically Aeolian vibrations—in overhead power transmission lines. These vibrations pose a significant threat to the structural integrity and operational lifespan of conductors, particularly in aging grid infrastructure. Existing mitigation strategies, such as passive dampers, often fall short in adaptability and effectiveness across varying environmental conditions. This work addresses these challenges by proposing a data-driven, sensor-efficient system capable of real-time dynamic state estimation and antinode tracking.

The study began by examining traditional mathematical models of conductor vibration, using an Euler-Bernoulli beam formulation under tension with an attached mass-damping system. While accurate, these models proved limited in their capacity to adapt to real-time disturbances or variations in system parameters.

To overcome this limitation, a machine learning-based approach was introduced. A Long Short-Term Memory (LSTM) neural network was trained to predict vibration behavior from sparse time-series input, effectively estimating the full vibration profile from limited sensor data. A curve-fitting procedure using cubic splines further enhanced the spatial resolution of the predictions.

A key contribution of this thesis is the application of Dynamic Mode Decomposition (DMD) for extracting low-dimensional representations of the system's behavior directly from empir-

ical data. To ensure numerical stability, Tikhonov Regularization was incorporated, yielding a reliable and stable transition matrix even under noisy or incomplete input conditions.

The framework was extended with real-time state estimation algorithms, namely the Kalman filter and particle filter, which enabled the reconstruction of unmeasured system states using model predictions and partial observations. These filters proved effective in dynamically tracking the state of the conductor under various configurations.

A significant innovation in this work is the sensor placement strategy based on pivoted QR decomposition of Proper Orthogonal Decomposition (POD) modes. This technique identified the most informative spatial points for sensor deployment, minimizing hardware requirements while preserving reconstruction accuracy.

Finally, the robustness of the system was further enhanced using compressive sensing, a method that allows accurate signal reconstruction from incomplete data. This approach proved effective in recovering lost or under-sampled vibration signals, ensuring the framework's resilience in practical deployment scenarios.

In combination, these contributions form an end-to-end pipeline for intelligent vibration monitoring in transmission infrastructure. The system leverages sparse sensing, machine learning, data-driven modeling, and signal recovery to provide a scalable, real-time solution with clear potential for real-world application. This thesis lays the foundation for future research into autonomous damping systems, adaptive control strategies, and broader implementations across smart grid technologies.

Bibliography

[1] Allied Market Research. Inspection robots market. https://www.alliedmarketresearch.com/inspection-robots-market-A08254, August 2021. Accessed on 2022-09-18.

[2] Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias, and Guowei Cai. A survey on inspecting structures using robotic systems. *International Journal of Advanced Robotic Systems*, 13(6):1729881416663664, 2016.

[3] American Society of Civil Engineers. 2021 infrastructure report card. https://infrastructurereportcard.org/wp-content/uploads/2020/12/Energy-2021.pdf, March 2021. Accessed on 2022-09-18.

[4] Andlinger Center for Energy and the Environment. Big but affordable effort needed for america to reach net-zero emissions by 2050, princeton study shows. https://www.princeton.edu/news/2020/12/15/big-affordable-effort-needed-america-reach-net-zero-emissions-2050-princeton-study, December 2020. Accessed on 2022-09-18.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002. doi: 10.1109/78.978374.

[6] Richard G. Baraniuk. Compressive sensing [lecture notes]. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007. doi: 10.1109/MSP.2007.4286571.

[7] O. Barry, J. W. Zu, and D. C. D. Oguamanam. Analytical and experimental investiga-

tion of overhead transmission line vibration. *Journal of Vibration and Control*, 21(14): 2825–2837, 2015.

[8] O. Barry, R. Long, and D. C. D. Oguamanam. Rational damping arrangement design for transmission lines vibrations: Analytical and experimental analysis. *Journal of Dynamic Systems, Measurement, and Control*, 139(5):051012, 2017.

[9] Oumar Barry and Mohammad Bukhari. On the modeling and analysis of an energy harvester moving vibration absorber for power lines. In *Dynamic Systems and Control Conference*, volume 58288, page V002T23A005. American Society of Mechanical Engineers, 2017.

[10] Oumar Barry, Donatus C. D. Oguamanam, and Der Chyan Lin. Aeolian vibration of a single conductor with a stockbridge damper. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 227(5):935–945, 2013.

[11] Oumar Barry, J. W. Zu, and D. C. D. Oguamanam. Nonlinear dynamics of stockbridge dampers. *Journal of Dynamic Systems, Measurement, and Control*, 137(6):061017, 2015.

[12] Oumar Rafiou Barry, Emadeddin Y. Tanbour, Nitish Kumar Vaja, and Hesham Tanbour. Asymmetric aeolian vibration damper, April 2018.

[13] Rafiou Oumar Barry. *Vibration Modeling and Analysis of a Single Conductor With Stockbridge Dampers*. Phd thesis, University of Toronto, 2014.

[14] Jason Brownlee. *Deep Learning for Time Series Forecasting*. Machine Learning Mastery, 2018.

[15] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings*

of the National Academy of Sciences (PNAS), 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

[16] Emmanuel J. Candes and Michael B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008. doi: 10.1109/MSP.2007.914731.

[17] Emmanuel J. Candes, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006. doi: 10.1109/TIT.2005.862083.

[18] Saifon Chaturantabut and Danny C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010. doi: 10.1137/090766498.

[19] Kyeong Ho Cho, Young Hoon Jin, Ho Moon Kim, Hyungpil Moon, Ja Choon Koo, and Hyouk Ryeol Choi. Multifunctional robotic crawler for inspection of suspension bridge hanger cables: Mechanism design and performance validation. *IEEE/ASME Transactions on Mechatronics*, 22(1):236–246, 2016.

[20] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52 (4):1289–1306, 2006. doi: 10.1109/TIT.2006.871582.

[21] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001. ISBN 978-0-387-95146-0.

[22] Jixiong Fei, Bin Lin, Shuai Yan, Mei Ding, Juliang Xiao, Jin Zhang, Xiaofeng Zhang, Chunhui Ji, and Tianyi Sui. Chatter mitigation using moving damper. *Journal of Sound and Vibration*, 410:49–63, 2017.

[23] Yuan Gao, Guangming Song, Songtao Li, Fushuai Zhen, Dabing Chen, and Aiguo Song.

LineSpyX: A power line inspection robot based on digital radiography. *IEEE Robotics and Automation Letters*, 5(3):4759–4765, 2020.

[24] Chiara Gazzola, Francesco Foti, Luca Martinelli, and Federico Perotti. An appraisal of modelling strategies for assessing aeolian vibrations of transmission lines. In *24th Conference of the Italian Association of Theoretical and Applied Mechanics, AIMETA 2019*, pages 1522–1534. Springer, 2020.

[25] Bertrand Godard, Suzanne Guerard, and Jean-Louis Lilien. Original real-time observations of aeolian vibrations on power-line conductors. *IEEE Transactions on Power Delivery*, 26(4):2111–2117, 2011.

[26] Suzanne Guerard, Bertrand Godard, and Jean-Louis Lilien. Aeolian vibrations on power-line conductors, evaluation of actual self damping. *IEEE Transactions on Power Delivery*, 26(4):2118–2122, 2011.

[27] Per Christian Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review*, 34(4):561–580, 1992. doi: 10.1137/1034115.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[29] Kunpeng Ji, Bin Liu, Jingchao Wang, Peng Li, Danyu Li, Lichun Zhang, Jingshan Han, and Yifeng Wang. Aeolian vibration and its suppression methods of long crossing span overhead electric transmission line. In *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*, pages 663–667. IEEE, 2019.

[30] Paul Kakou and Oumar Barry. Simultaneous vibration reduction and energy harvesting of a nonlinear oscillator using a nonlinear electromagnetic vibration absorber-inerter. *Mechanical Systems and Signal Processing*, 156:107607, 2021.

[31] Paul Kakou, Mohammad Bukhari, Jiamin Wang, and Oumar Barry. On the vibration suppression of power lines using mobile damping robots. *Engineering Structures*, 239: 112312, 2021.

[32] Paul-Camille Kakou. *Towards A Mobile Damping Robot For Vibration Reduction of Power Lines*. Phd thesis, Virginia Tech, 2021.

[33] Paul-Camille Kakou and Oumar Barry. Toward a mobile robot for vibration control and inspection of power lines. *ASME Letters in Dynamic Systems and Control*, 2(1): 011001, 2021.

[34] Y. D. Kubelwa, R. C. Loubser, and P. Moodley. Experimental investigations of bending stresses of acsr conductors due to aeolian vibrations. In *CIGRE Science & Engineering*, volume 9, pages 1286–1146, 2017. Innovation in the Power Systems Industry.

[35] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[36] Eduardo José Lima, Marcelo Henrique Souza Bomfim, and Miguel Augusto de Miranda Mourão. Polibot–power lines inspection robot. *Industrial Robot: An International Journal*, 2018.

[37] M. L. Lu and J. K. Chan. An efficient algorithm for aeolian vibration of single conductor with multiple dampers. *IEEE Transactions on Power Delivery*, 22(3):1822–1829, 2007.

[38] Krithika Manohar, Steven L. Brunton, J. Nathan Kutz, and Bingni W. Brunton. Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, 38(3):63–86, 2018. doi: 10.1109/MCS.2018.2810460.

[39] B. M. Miller and E. Ya. Rubinovich. Regularization of a generalized kalman filter. *Mathematics and Computers in Simulation*, 39(1–2):87–108, 1995. doi: 10.1016/0378-4754(95)00024-R.

[40] Hyun Myung, Yang Wang, S. C. Kang, and XiaoQi Chen. Survey on robotics and automation technologies for civil infrastructure, 2014.

[41] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 60 (1):123–164, 2016. doi: 10.1137/16M1056200.

[42] PG&E. Electric transmission line inspection and preventive maintenance program. https://www.pge.com/pge_global/common/pdfs/safety/emergency-preparedness/natural-disaster/wildfires/wildfire-mitigation-plan/reference-docs/TD-1001S.pdf, January 2020. Accessed on 2022-09-18.

[43] Andrew Phillips, Eric Engdahl, Drew McGuire, Mark Major, and Glynn Bartlett. Autonomous overhead transmission line inspection robot (ti) development and demonstration. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 94–95. IEEE, 2012.

[44] Preformed Line Products. Aeolian vibration basics. http://www.preformed.com/images/pdfs/Energy/Transmission/Motion_Control/VORTX_Vibration_Damper/EN-ML-1007-4AeolianViBook.pdf, October 2013. Accessed on 2022-09-18.

[45] President's Council of Economic Advisers. Economic benefits of increasing electric grid resilience to weather outages. https://www.energy.gov/sites/prod/files/2013/08/f2/Grid%20Resiliency%20Report_FINAL.pdf, March 2017. Accessed on 2022-09-18.

[46] A. Rezaei and M. H. Sadeghi. Analysis of aeolian vibrations of transmission line conductors and extraction of damper optimal placement with a comprehensive methodology. *International Journal of Engineering*, 32(2):328–337, 2019.

[47] Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009. doi: 10.1017/S0022112009992059.

[48] Morteza Sadeghi and Aryo Rezaei. Aeolian vibrations of transmission line conductors with more than one damper. *International Journal of Engineering*, 28(10):1515–1524, 2015.

[49] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217.

[50] S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401. IEEE, 2018. doi: 10.1109/ICMLA.2018.00227.

[51] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006. ISBN 978-0-471-70858-2.

[52] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN 978-0-262-20162-9.

[53] A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, and A. G. Yagola. Algorithms and programs for solving linear ill-posed problems. In *Numerical Methods for the Solution of Ill-Posed Problems*, pages 97–161. Springer Netherlands, 1995. doi: 10.1007/978-94-015-8480-7_5.

[54] US EPRI and EDF R&D. Profiling and mapping of intelligent grid r&d programs. Technical report, US EPRI, Palo Alto, CA, 2006.

[55] Utility Products. Drones for power line inspections. https://www.utilityproducts.com/line-construction-maintenance/article/16003823/drones-for-power-line-inspections, January 2019. Accessed on 2022-09-18.

[56] N. K. Vaja, O. R. Barry, and E. Y. Tanbour. On the modeling and analysis of a vibration absorber for overhead powerlines with multiple resonant frequencies. *Engineering Structures*, 175:711–720, 2018.

[57] Nitish Kumar Vaja, Oumar Barry, and Brian DeJong. Finite element modeling of stockbridge damper and vibration analysis: Equivalent cable stiffness. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 58226, page V008T12A012. American Society of Mechanical Engineers, 2017.

[58] Y. R. Wang and C. Y. Lo. Design of hybrid dynamic balancer and vibration absorber. *Shock and Vibration*, 2014, 2014. doi: 10.1155/2014/149386.

[59] Zhaoyang Wang, Qiang Gao, Jianbin Xu, and Dahua Li. A review of uav power line inspection. In Liang Yan, Haibin Duan, and Xiang Yu, editors, *Advances in Guidance, Navigation and Control*, pages 3147–3159. Springer Singapore, 2022. ISBN 978-981-15-8155-7.

[60] Liman Yang, Chenyao Fu, Yunhua Li, and Lianming Su. Survey and study on intelligent monitoring and health management for large civil structure. *International Journal of Intelligent Robotics and Applications*, 3(3):239–254, 2019.

Appendices

Appendix A

Neural Network Code

```
import numpy as np
import pandas as pd
import tensorflow as tf

from keras.layers import Input, Dense, Dropout, Lambda,
concatenate, Conv2D, MaxPooling2D, Flatten, LSTM, BatchNormalization,
MultiHeadAttention, Conv1D, GlobalAveragePooling1D, LayerNormalization,
GlobalMaxPooling1D
from keras.models import Model, load_model
from keras.optimizers import Adam
from keras.regularizers import l2
from keras import backend as K
from scipy.sparse import coo_matrix
import math
from scipy.fftpack import fft, ifft
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
from keras.models import Sequential
```

```
19 import pysindy as ps
20 from scipy.linalg import eigvalsh, eigh
21
22 filename = "21_points.csv"
23
24 ## Reading Data
25 data = pd.read_csv(filename, sep = ",")
26 data = data.to_numpy()*10**4
27 data = data[10000:,]
28
29 ## Preparing Data
30 data_train = data[:round(data.shape[1]*0.8),:]
31 data_test = data[round(data.shape[1]*0.8):,:]
32
33
34 ## Model Structure
35 inputs = Input(shape =[(x_train.shape[1]),1])
36 x = inputs
37 x = LayerNormalization(epsilon = 0.001)(x)
38 x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
39 x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
40 x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
41 x = LSTM(64, activation = 'tanh', return_sequences = 'true')(x)
42 x = Flatten()(x)
43 x = Dropout(0.2)(x)
44 x = Dense(21, activation = 'tanh')(x)
```

```
45
46 res = x + Flatten()(inputs)
47 x = Dense(512, activation = 'tanh')(res)
48 x = Dropout(0.2)(x)
49 x = Dense(256, activation = 'tanh')(x)
50 outputs = Dense(21, activation = 'linear')(x)
51
52 model = Model(inputs,outputs)
53
54 ## Compile Model
55 model.compile(optimizer='adam', loss = custom_loss, metrics=['accuracy'])
56
57 ## Early Stop
58 early_stop = tf.keras.callbacks.EarlyStopping(monitor = 'loss',
59 patience = 10, mode = 'min', start_from_epoch= 25)
60
61 ## Train the Model
62 history = model.fit(x_train, y_train, epochs=2000, batch_size = 64,
63 verbose= 1, validation_split = 0.2, callbacks = [early_stop])
64
```

Appendix B

Dynamic Mode Decomposition Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#######################################################
file_index = 3
#######################################################

def read_data(file_index):
    data = pd.read_csv(filenames[file_index], sep = ",")
    data = data.to_numpy()*10**4
    data = data[10000:,]
    return data

#######################################################
#######################################################

filenames = ['Null','21_points.csv', '101_points.csv', 'm_10_L_2.csv']
```

```
data = read_data(file_index)
dmd_data = data.T
n_states = data.shape[1] #Number of states in the system

#########################################################

n_observed = 51 #Number of measurable states starting from X_1 to X_n_observed

#########################################################

n_steps = 1000 #Number of time steps under consideration
start = 1000 #Starting point of data, to avoid transience if applicable
q = 0.001 #Process Noise
r = 0.01 #Measurement noise
p = 1000 #Initial Covariance
epsilon = 1 #Regularization for Kalman Filter
lambda_reg = 0.1 #Regularization for state-transition
plt_index = 500 #Plot index for Kalman Filter
#########################################################

order = 50 #Lag

#########################################################
start = 1000


```

```
45 ######################################################

46

47 def dmd_ekf():
48     if n_states < n_observed:
49         raise ValueError('Observable states are more than total number of states')
50     A_dmd, X_prime, X = A_from_DMD()
51     print('System after DMD:')
52     print(' ')
53     plot_eig(A_dmd)
54     print('System after DMD with regularization:')
55     print(' ')
56     print('Lambda: ' + str(lambda_reg))
57     A_reg = state_transition(X_prime, X, lambda_reg)
58     plot_eig(A_reg)
59     print('Performance of Kalman Filter: ')
60     print(' ')
61     print('Total number of states: ' + str(n_states))
62     print('Number of observable states: ' + str(n_observed))
63     print('Epsilon: ' + str(epsilon))
64     print('Plot Index (time step): ' + str(plt_index))
65     kalman_filter(dmd_data, n_states, n_observed, n_steps, A_reg, epsilon, plt_index)

66

67 ######################################################

68

69 def preview_dmd():
70     new_test = start + 1000
```

```
71      newX = dmd_data[:,new_test - order:new_test]
72      newX_prime = np.dot(A_dmd,newX)
73      ## Testing
74      X_dmd = newX_prime[:,order - 1]
75      act_X =dmd_data[:,new_test]
76      fig_dmd, ax_dmd = plt.subplots(nrows = 1, ncols = 1)
77      ax_dmd.grid(True, linestyle = '--', linewidth = 0.5)
78      ax_dmd.plot(np.arange(n_states), act_X, label = 'Actual Data')
79      ax_dmd.plot(np.arange(n_states),X_dmd, label = 'Data-Driven Model',
80      linestyle = 'dashed', color = 'r', dashes = (5,5))
81      ax_dmd.legend()
82
83  ##########################################################
84
85  def A_from_DMD():
86      X = dmd_data[:,start:start + order]
87      X_prime = dmd_data[:,start + 1:start + order + 1]
88      X_pinv = np.linalg.pinv(X)
89      A_dmd = np.dot(X_prime,X_pinv)
90      return A_dmd, X_prime, X
91
92  ##########################################################
93
94  def plot_eig(A):
95      A_eig, _ = np.linalg.eig(A)
96      A_eig = np.array(A_eig)
```

```
97
98      imag_max = np.max(np.abs(np.imag(A_eig)))
99      print("Max Imaginary Part: " + str(imag_max))
100
101     real_max = np.max(np.abs(np.real(A_eig)))
102     print("Max Real Part: " + str(real_max))
103
104     fig, ax = plt.subplots(figsize = (6,6))
105
106     theta = np.linspace(0, 2*np.pi, 100)
107     x = np.cos(theta)
108     y = np.sin(theta)
109     ax.plot(x, y, 'k--', linewidth = 1.5, label = 'Unit Circle')
110     inside_eig = A_eig[np.abs(A_eig) <= 1]
111     outside_eig = A_eig[np.abs(A_eig) > 1]
112
113     print("Total number of eigenvalues: " + str(A_eig.shape[0]))
114
115     print("Eigenvalues inside unit circle: " + str(inside_eig.shape[0]))
116
117     if not outside_eig.size > 0:
118         print("No Eigenvalues outside unit circle")
119         print("System is stable")
120
121     else:
122         print("Eigenvalues outside the unit circle: " + str(outside_eig.shape[0]))
```

```
123         print("System is unstable")
124
125     marker_size = 5
126     ax.plot(np.real(inside_eig), np.imag(inside_eig), 'bo',
127     markersize= marker_size, label='Eigenvalues (Inside)')
128     ax.plot(np.real(outside_eig), np.imag(outside_eig), 'ro',
129     markersize= marker_size, label='Eigenvalues (Outside)')
130
131     ax.set_aspect('equal')
132
133     limit = np.max([[imag_max], [real_max]], initial = 1.2)
134     if not limit == 1.2:
135         limit = limit + 500
136     ax.set_xlim(-limit, limit)
137     ax.set_ylim(-limit, limit)
138
139     ax.grid(True, linestyle = '--', linewidth = 0.5)
140
141     ax.set_xlabel('Real')
142     ax.set_ylabel('Imaginary')
143     ax.set_title('Eigenvalues')
144
145     ax.legend()
146     plt.tight_layout()
147     plt.show()
148
```

```
149 ########################################################

150

151 def state_transition(X, Y, lambda_reg):

152

153     X_reg = X @ X.T + lambda_reg*np.eye(X.shape[0])

154     X_pinv_reg = np.linalg.pinv(X_reg) @ X

155     A_reg = Y @ X_pinv_reg.T

156     return A_reg

157

158 ########################################################

159

160 def kalman_filter(dmd_data, n_states, n_observed, n_steps, A_reg,

161 epsilon, plt_index = 750):

162     Q = np.eye(n_states)*q

163     R = np.eye(n_observed)*r

164     P = np.eye(n_states)*p

165     w = np.random.multivariate_normal(np.zeros(n_states), Q, 1).T

166     v = np.random.multivariate_normal(np.zeros(n_observed), R, 1).T

167

168     H = np.zeros((n_observed, n_states))

169     for i in range(n_observed):

170         H[i,i] = 1

171

172     x_hat = dmd_data[:,start - 1]

173     z = dmd_data[:n_observed,start:start + n_steps]

174     A = A_reg
```

```
175     estimated_states = np.zeros((n_states,n_steps))
176     for k in range(n_steps):
177         x_hat_pred = np.dot(A_reg,x_hat)
178         P_pred = np.dot(np.dot(A,P),A.T) + Q + np.eye(n_states)*epsilon
179         if np.trace(P_pred) > 1e10:
180             print(k)
181             P_pred = np.eye(n_states) * p
182         y = z[:,k] - np.dot(H,x_hat_pred)
183         S = np.dot(np.dot(H,P_pred),H.T) + R + np.eye(n_observed)*epsilon
184         det_S = np.linalg.det(S)
185         #if det_S < 1e-6:
186             #print(k)
187             #print('Near Singular Matrix detected')
188             #raise ValueError('Near Singular Matrix detected')
189         K = np.dot(np.dot(P_pred,H.T),np.linalg.pinv(S + np.eye(S.shape[0])*epsilon))
190         x_hat = x_hat_pred + np.dot(K,y)
191         P = np.eye(n_states) - np.dot(np.dot(K,H),P_pred)
192         estimated_states[:,k] = x_hat
193
194     fig_kf, ax_kf = plt.subplots(nrows = 1, ncols = 1)
195     ax_kf.grid(True, linestyle = '--', linewidth = 0.5)
196     ax_kf.plot(np.arange(n_states), dmd_data[:,start + plt_index],
197     label = 'Actual Data')
198     ax_kf.plot(np.arange(n_states),estimated_states[:,plt_index],
199     label = 'Data-Driven Model', linestyle = 'dashed', color = 'r', dashes = (5,5))
```

```
200     ax_kf.legend()
```

Appendix C

Particle Filter Code

```
import time
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import ipywidgets as widgets
from IPython.display import display
from scipy.stats import multivariate_normal
import scipy.stats
import pandas as pd
from pykalman import KalmanFilter
from pfilter import ParticleFilter, gaussian_noise,
squared_error, independent_sample
start = 10000
init_state = data[:,start]
num_particles = 100
num_states = 21
sensor_std_dev = 0.5
sensor_pos = [2, 4, 6, 8, 10, 12, 14, 16, 18, 19]
```

```
19 time_steps = 100
20
21 def sensor(actual_state):
22     return actual_state[sensor_pos]
23
24 def predict_particles(particles, num_particles):
25     for i in range(num_particles):
26         particles[:,i] = np.dot(A,particles[:,i])
27
28     return particles
29
30 def update_weights(particles, weights, measurement, sensor_std_dev):
31     log_weights = np.zeros(num_particles)
32     for i in range(num_particles):
33         log_weight = 0.0
34         for j in range(len(measurements)):
35             sensor_measurement = measurements[j]
36             log_weight -= (particles[j, i] - sensor_measurement) ** 2
37             / (2 * sensor_std_dev ** 2)
38         log_weights[i] = log_weight
39     max_log_weight = np.max(log_weights)
40     weights = np.exp(log_weights - max_log_weight)
41     weights /= np.sum(weights)
42
43
44 def resample_particles(particles, weights, num_particles):
```

```
45      cumulative_sum = np.cumsum(weights)
46      cumulative_sum[-1] = 1
47      indexes = np.searchsorted(cumulative_sum, np.random.rand(num_particles))
48      return particles[:,indexes], np.ones(num_particles) / num_particles
49
50  def estimate_state(particles):
51      return np.mean(particles, axis = 1)
52
53
54  def init_particles(num_particles,num_states):
55      return data[:,start:start+num_particles]
56
57  particles = init_particles(num_particles,num_states)
58  weights = np.ones(num_particles)/num_particles
59  state_estimate = np.zeros((num_states, time_steps))
60
61  for t in tqdm(range(time_steps), desc="Processing", unit = "Iteration"):
62
63      tt = time.time()
64      actual_state = data[:,start+t]
65      measurements = sensor(actual_state)
66      particles = predict_particles(particles,num_particles)
67      update_weights(particles, weights, measurements,sensor_std_dev)
68      particles, weights = resample_particles(particles, weights, num_particles)
69
```

```
70      state_estimate[:,t] = estimate_state(particles)
```

Appendix D

Compressive Sensing

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import scipy.optimize as spopt
import scipy.fftpack as spfft
import scipy.ndimage as spimg
import cvxpy as cvx

def read_data(filename):
    data = pd.read_csv(filename, sep = ",")
    data = data.to_numpy()*10**4
    return data

filename = '101_points.csv'
data = read_data(filename)
## Defining the signal and temporal space
num_sig = data.shape[1]
```

```
num_time_steps = data.shape[0]
time_step = 0.001
start_time = 0
end_time = time_step*num_timesteps
t = np.linspace(start_time, end_time,num_time_steps)

## Plotting the signals
sig_index = 50
zoom_lim = 1500
fig, axs = plt.subplots(2, figsize = (20,10))
axs[0].plot(t,data[:,sig_index])
axs[0].set_title("Original Signal at point " + str(sig_index))
axs[0].grid()
axs[1].plot(t[:zoom_lim],data[:zoom_lim,sig_index])
axs[1].set_title("Zoomed in original signal")
axs[1].grid()

## Truncating the signal
data2 = data[:5000,:]

## Sampling the signal
samp = 300
samp_ind = np.random.choice(data2.shape[0], samp, replace = False)
samp_ind.sort()
t_cs = t[samp_ind]
y_cs = data2[samp_ind,sig_index]
```

```
45
46 fig, axs = plt.subplots(1,2, figsize = (20,5))
47 axs[0].plot(t[:500],data2[:500,sig_index])
48 axs[0].plot(t_cs,y_cs,'ro')
49 axs[0].set_xlim((0, t[500]))
50 axs[0].grid()
51 axs[0].set_title('Original Signal vs Sample')
52 axs[1].plot(t_cs,y_cs)
53 axs[1].grid()
54 axs[1].set_title('Signal from the sample')
55 axs[1].set_xlim((0, t[500]))
56
57 A_cs = spfft.idct(np.identity(5000), norm = 'ortho', axis = 0)
58 A_cs = A_cs[samp_ind]
59
60 vx = cvx.Variable(5000)
61 objective = cvx.Minimize(cvx.norm(vx,1))
62 constraints = [A_cs*vx == y_cs]
63 problem = cvx.Problem(objective, constraints)
64 result = problem.solve(verbose = False)
65
66 x = np.array(vx.value)
67 x = np.squeeze(x)
68 sig = spfft.idct(x, norm = 'ortho', axis = 0)
69
70 fig, axs = plt.subplots(2, figsize = (20,10))
```

```
71 axs[0].plot(t[:5000],sig)
72 axs[0].grid()
73 axs[0].set_xlim(0, t[1000])
74 axs[0].set_title('Reconstructed signal')
75 axs[1].plot(t[:5000],data2[:,sig_index])
76 axs[1].grid()
77 axs[1].set_xlim(0, t[1000])
78 axs[1].set_title('Original signal')
79 fig.tight_layout()
80
81 ## System details
82 plt_index = 500
83 N = data.shape[1]
84 ## Defining the x space
85 x_space = np.arange(data.shape[1])
86 ## Sampling the signal I guess
87 samp = 50
88 samp_ind = np.random.choice(data.shape[1],samp, replace = False)
89 samp_ind.sort()
90 x_cs = x_space[samp_ind]
91 y_cs = data[500, samp_ind]
92
93 A_cs = spfft.idct(np.identity(N), norm = 'ortho', axis = 0)
94 A_cs = A_cs[samp_ind]
95
96 vx = cvx.Variable(N)
```

```
97  objective = cvx.Minimize(cvx.norm(vx,1))
98  constraints = [A_cs*vx == y_cs]
99  problem = cvx.Problem(objective, constraints)
100 result = problem.solve(verbose = False)
101
102 x = np.array(vx.value)
103 x = np.squeeze(x)
104 sig = spfft.idct(x, norm = 'ortho', axis = 0)
105
106 fig, axs = plt.subplots(2, figsize=(20,10))
107 axs[0].plot(x_space,sig)
108 axs[0].set_title('Reconstucted profile')
109 axs[0].grid()
110 axs[1].plot(np.arange(data.shape[1]),data[10005])
111 axs[1].set_title('Original profile')
112 axs[1].grid()
113
```

Appendix E

Neural Network Models

This section documents the different Neural Network architectures that have been explored for this thesis.

```
model.add(LSTM(21, input_shape = [(x_train.shape[1]),1],
activation = 'tanh', return_sequences = 'true'))
model.add(GlobalAveragePooling1D(data_format='channels_last'))
model.add(Dense(128, activation = 'tanh'))
model.add(Dense(512, activation = 'tanh'))
model.add(Dropout(0.2))
model.add(Dense(256, activation = 'tanh'))
model.add(Dense(21, activation = 'linear'))
```

```
model.add(LSTM(21, input_shape = [(x_train.shape[1]),1],
activation = 'tanh', return_sequences = 'true'))
model.add(GlobalMaxPooling1D(data_format='channels_last'))
model.add(Dense(128, activation = 'tanh'))
model.add(Dense(512, activation = 'tanh'))
model.add(Dropout(0.2))
```

```
model.add(Dense(256, activation = 'tanh'))
model.add(Dense(21, activation = 'linear'))
```

```
model.add(LSTM(21, input_shape = [(x_train.shape[1]),1],
activation = 'tanh', return_sequences = 'true'))
model.add(Conv1D(512,5,padding='same', activation = 'tanh'))
model.add(GlobalMaxPooling1D(data_format='channels_last'))
model.add(Flatten())
model.add(Dense(128, activation = 'tanh'))
model.add(Dense(512, activation = 'tanh'))
model.add(Dropout(0.2))
model.add(Dense(256, activation = 'tanh'))
model.add(Dense(21, activation = 'linear'))
```

```
inputs = Input(shape =[(x_train.shape[1]),1])
x = inputs
x = LayerNormalization(epsilon = 0.001)(x)
x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
x = LSTM(64, activation = 'tanh', return_sequences = 'true')(x)
x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(21, activation = 'tanh')(x)
```

```
11
12 res = x + Flatten()(inputs)
13 x = Dense(512, activation = 'tanh')(res)
14 x = Dropout(0.2)(x)
15 x = Dense(256, activation = 'tanh')(x)
16 outputs = Dense(21, activation = 'linear')(x)
```

```
1  inputs = Input(shape =[(x_train.shape[1]),1])
2  x = inputs
3
4  for i in range(3):
5      x = LayerNormalization(epsilon = 0.001)(x)
6      x = MultiHeadAttention(key_dim = 256, num_heads = 5, dropout = 0.2)(x,x)
7      x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
8      x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
9      x = LSTM(21, activation = 'tanh', return_sequences = 'true')(x)
10
11 x = Flatten()(x)
12 x = Dense(21, activation = 'tanh')(x)
13 res = x + Flatten()(inputs)
14 x = Dense(512, activation = 'tanh')(res)
15 x = Dropout(0.2)(x)
16 x = Dense(256, activation = 'tanh')(x)
17 outputs = Dense(21, activation = 'linear')(x)
```

```
inputs = Input(shape =[(x_train.shape[1]),1])
x = inputs


x = LayerNormalization(epsilon = 0.001)(x)
x = MultiHeadAttention(key_dim = 256, num_heads = 5, dropout = 0.2)(x,x)
x = Conv1D(512,21,padding='same', activation = 'tanh')(x)
x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
x = LSTM(21, activation = 'tanh')(x)

#x = Flatten()(x)
#x = Dense(21, activation = 'tanh')(x)
res = x - Flatten()(inputs)
x = Dense(512, activation = 'tanh')(res)
x = Dropout(0.2)(x)
x = Dense(256, activation = 'tanh')(x)
outputs = Dense(21, activation = 'linear')(x)
```

```
inputs = Input(shape =[(x_train.shape[1]),1])
x = inputs


x = LayerNormalization(epsilon = 0.001)(x)
x = MultiHeadAttention(key_dim = 256, num_heads = 5, dropout = 0.2)(x,x)
```

```
x = Conv1D(512,21,padding='same', activation = 'tanh')(x)
x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
x = LSTM(21, activation = 'tanh')(x)

#x = Flatten()(x)
#x = Dense(21, activation = 'tanh')(x)
res = x + Flatten()(inputs)
x = Dense(512, activation = 'tanh')(res)
x = Dropout(0.2)(x)
x = Dense(256, activation = 'tanh')(x)
outputs = Dense(21, activation = 'linear')(x)
```

```
inputs = Input(shape =[(x_train.shape[1]),(x_train.shape[2])])
x = inputs


x = LayerNormalization(epsilon = 0.001)(x)
x = MultiHeadAttention(key_dim = 256, num_heads = 5, dropout = 0.2)(x,x)
x = Conv1D(512,5,padding='same', activation = 'tanh')(x)
x = LSTM(128, activation = 'tanh', return_sequences = 'true')(x)
x = LSTM(x_train.shape[1]*x_train.shape[2], activation = 'tanh', return_sequences = 'tru
x = GlobalMaxPooling1D(data_format='channels_last')(x)

res = x + Flatten()(inputs)
x = Dense(512, activation = 'tanh')(res)
```

```
x = Dropout(0.2)(x)
x = Dense(256, activation = 'tanh')(x)
outputs = Dense(21, activation = 'linear')(x)

```