Machine learning for nonlinear materials characterization and modeling

Daniel E. Shea

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

J. Nathan Kutz, Chair

Steven L. Brunton, Chair

David S. Ginger

Fumio S. Ohuchi

Program Authorized to Offer Degree:
Materials Science and Engineering

University of Washington

Abstract

Machine learning for nonlinear materials
characterization and modeling

Daniel E. Shea

Co-Chairs of the Supervisory Committee:
Robert Bolles and Yasuko Endo Professor of Applied Mathematics
J. Nathan Kutz
Department of Applied Mathematics

James B. Morrison Endowed Career Development Professor of Mechanical Engineering
Steven L. Brunton
Department of Mechanical Engineering

Materials scientists and engineers broadly aim to study materials by analyzing their structures, performance, properties, and synthesis methods using a variety of characterization techniques. This thesis aims to develop broadly applicable data-driven techniques to advance the study of materials by improving characterization and modeling of nonlinear materials. Nonlinear materials are generally challenging to understand because of the difficulty associated with solving the relevant governing differential equations. Furthermore, many systems in materials science and engineering are governed by boundary value problems wherein certain conditions are specified at the points within or boundaries of the system. In this work, we develop two data-driven modeling approaches for boundary value problems (BVPs) involving nonlinear differential equations and one characterization technique for time-frequency analysis of a nonlinear phase evolution system. The data-driven modeling approaches can be used to understand the underlying physics, enable predictive modeling of the system, *and* are broadly applicable to any BVPs. The time-frequency analysis technique improves

the time-frequency resolution of traditional techniques, enables analysis of nonstationary time series signals, and can be used on any multimodal nonstationary signal. Further, it is extremely useful for analyzing cantilever-based imaging modalities that are extremely common in materials science such as atomic force microscopy.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables ix

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Organization 3
1.3 Bibliographic Note 4

Chapter 2: SINDy-BVP: Sparse identification of nonlinear dynamics for boundary value problems 5
2.1 Introduction 5
2.2 Background 9
2.3 Methods 12
2.4 Computational Results 22
2.5 Discussion 31

Chapter 3: DeepGreen: Deep Learning of Green's Functions for Nonlinear Boundary Value Problems 41
3.1 Introduction 42
3.2 Deep Autoencoders for Linearizing BVPs 45
3.3 Results 50
3.4 Conclusion 61

Chapter 4: Extraction of instantaneous frequencies and amplitudes in non-stationary time-series data 66
4.1 Introduction 66
4.2 Methods 70

4.3 Results 77
4.4 Conclusion 91

Chapter 5: Conclusion 93

LIST OF FIGURES

Figure Number Page

2.1 SINDy-BVP studies steady-state systems subjected to a forcing function. One simple example system is a beam clamped at both ends subjected to a static load (a). The beam deflects (b) in response to the load, and the forcing function and deflection are used for data-driven modeling via SINDy-BVP to learn the parametric coefficients (c) in the governing operator. The coefficients $p(x)$ and $q(x)$ vary spatially. The coefficients are directly related to the beam's spatially-varying mechanical properties. The grey boxes in (d) indicate that error can occur in the learned coefficients near the boundaries. 34

2.2 Overview of constructing the data sets and regression for each discrete spatial point in the data set. Part (a) shows a collection of trials, where different forcings ($f_j(x)$) are applied to a system yielding different responses ($u_j(x)$). A matrix containing a library of candidate terms $\mathbf{\Theta}(\mathbf{U}, \mathbf{F})$ is produced for each trial, where the rows are each a spatial point x_k and each column contains a candidate function, as seen in part (b). A sliding procedure is used to select rows of a single x_k from the libraries in part (b) to produce an aggregated library $\mathbf{\Theta}^{(k)}$ for each x_k in the data set. In (c), the regression is performed for each x_k to produce a vector of the parametric coefficients $p(x)$ and $q(x)$ at each x_k. The regression in (c) is a Ridge regression algorithm. However, a sparsity constraint is applied by an iterative thresholding and grouping mechanism of the algorithm. 35

2.3 Summary of the models and operators studied with SINDy-BVP. The center column shows three example trials from the training data set ($u_1(x)$, $u_2(x)$, and $u_3(x)$). The solutions are all of $\mathcal{O}(1)$. The final column shows the parametric coefficients in the operator, as well as the inferred parameters for clean data. Coefficients $p(x)$ and $q(x)$ are plotted with an offset, with markers every 30 points. 37

2.4 SINDy-BVP succeeds at operator identification in signals with high SNR. A collection of trials with varying SNR is used for data-driven operator identification. The plots in (a) and (b), from top to bottom, are the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, Poisson, and Euler-Bernoulli beam models. Relative loss (Equation 2.10) and a count of spurious terms in the identified model are quantified. In (a), 200 trials are used for regression and the SNR is varied. In (b), the number of trials is varied while holding a constant SNR of 100. . 38

2.5 Parametric coefficient estimation in noisy data. With a known operator L, SINDy-BVP can estimate the parametric coefficients. Coefficient error is used to compare the effect of varying SNR (a) and the number of trials used as input data (b). In (a), the number of trials used as input data are held constant at 200 and in (b) the SNR is fixed at 100. The plots in (a) and (b), from top to bottom, are the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, Poisson, and Euler-Bernoulli beam models. 39

2.6 Determination of correct order equation to use for building the data-driven model. This example is the Euler-Bernoulli beam theory, which should use the left hand side term $d^4u(x)/dx^4$ to build the correct model. The model which best captures the relationship $Lu(x) = f(x)$ is determined from the validation error $\|L[\mathbf{U}_j] - \mathbf{F}_j\|_2$ from a test data set. The fourth order model ($d^4u(x)/dx^4$) exhibits the lowest error. . 40

3.1 DeepGreen solves nonlinear BVPs by identifying the Green's Function of the nonlinear problem using a deep learning approach with a dual autoencoder architecture. An nonhomogenous linear BVP can be solved using the Green's function approach, but a nonlinear BVP cannot. DeepGreen transforms a nonlinear BVP to a linear BVP, solves the linearized BVP, and then inverse transforms the linear solution to solve the nonlinear BVP. 45

3.2 DeepGreen architecture. Two autoencoders learn invertible coordinate transformations that linearize a nonlinear boundary value problem. The latent space is constrained to exhibit properties of a linear system, including linear superposition, which enables discovery of a Green's function for nonlinear boundary value problems. 48

3.3 Learning curve. This is a typical learning curve for the DeepGreen architecture. The vertical dashed line indicates where the training procedure transitions from autoencoders-only (only $\mathcal{L}_1$ and $\mathcal{L}_2$) to a full-network training procedure (all losses). 51

3.4 Latent space representations $\mathbf{v}_k$ and $\mathbf{f}_k$. The autoencoder transformation $\boldsymbol{\psi}_u$ encodes $\mathbf{u}_k$ to the latent space, producing the vector $\mathbf{v}_k$ (orange). The forcing vector $\mathbf{F}_k$ is transformed by $\boldsymbol{\psi}_F$ to the encoded vector $\mathbf{f}_v$ (blue). 52

3.5 Visualized operator and Green's function. Discovered Green's function $\mathbf{G} = \mathbf{L}^{-1}$ and corresponding linear operator $\mathbf{L}$. 52

3.6 Model predictions on test data. The top row shows the true solution $\mathbf{u}_k(x)$ and the solution predicted by the network given the forcing $\mathbf{F}_k(x)$ using the Green's function $\mathbf{G}$. The bottom row shows the true forcing function $\mathbf{F}_k(x)$ compared to the forcing computed by applying the operator $\mathbf{L}$ to the solution $\mathbf{u}_k$. Three columns show the best, mean, and worst case samples as evaluated by the sum of normalized $\ell 2$ reconstruction errors. 54

3.7 Model predictions on cubic Helmholtz forced system. The top row shows the true solution $\mathbf{u}_k(x)$ and the solution predicted by the network given the forcing $\mathbf{F}_k(x)$ using the Green's function $\mathbf{G}$. The bottom row shows the true forcing function $\mathbf{F}_k(x)$ compared to the forcing computed by applying the operator $\mathbf{L}$ to the solution $\mathbf{u}_k$. Three columns show the best, mean, and worst case samples as evaluated by the sum of normalized $\ell 2$ reconstruction errors. 56

3.8 Model performance summary. Distribution of loss values are shown for every sample in the test data set. Model loss functions are minimized during training, making them a natural metric to use for summarizing performance. 57

3.9 Model predictions for the (a) best and (b) worst examples from test data with Gaussian and cosine forcings. In both (a) and (b), the top row shows the true solution $\mathbf{u}(\mathbf{x})$, the predicted solution using the Green's function, and the difference between the true and predicted solution. The bottom row shows the true forcing function $\mathbf{F}(\mathbf{x})$, the predicted forcing function, and the difference between the true and predicted forces. In order to account for the difference in scale between $\mathbf{u}(\mathbf{x})$ and $\mathbf{F}(\mathbf{x})$, the differences are scaled by the infinity norm of the true solution or forcing function (Difference = (True − Predicted)/$||\text{True}||_\infty$). 63

3.10 Model predictions for the (a) best and (b) worst examples from test data with cubic polynomial forcings. In both (a) and (b), the top row shows the true solution $\mathbf{u}(\mathbf{x})$, the predicted solution using the Green's function, and the difference between the true and predicted solution. The bottom row shows the true forcing function $\mathbf{F}(\mathbf{x})$, the predicted forcing function, and the difference between the true and predicted forces. In order to account for the difference in scale between $\mathbf{u}(\mathbf{x})$ and $\mathbf{F}(\mathbf{x})$, the differences are scaled by the infinity norm of the true solution or forcing function (Difference = (True − Predicted)/$||$True$||_\infty$). . . . 64

3.11 Two-dimensional Poisson model performance summary. Distribution of loss values are shown for every sample in the test data set. Model loss functions are minimized during training, making them a natural metric to use for summarizing performance. 65

4.1 **Comparison between STFT and NFMD methods.** The top row shows an input signal, the second row shows a windowing function, and the third row shows the windowed signal. The fourth row provides the output of the decomposition. STFT computes the Fourier transform of a convolution of the windowed function, $\Psi(t)$. The example shown uses a Gaussian windowing function, which is commonly referred to as the Gabor transform. The bottom panel for STFT shows the FFT of the signal with real and imaginary components of the Fourier transform. The NFMD analyzes a segment of the signal, and fits a finite number of Fourier modes to the signal segment. The bottom panel for NFMD shows the two modes fit to the signal segment. 74

4.2 **Example signal for demonstrating NFMD.** The true signal is presented with the short time Fourier transform and the power spectral density (PSD). The signal has three maxima that appear in the PSD and STFT, indicating that there are likely three modes to be considered. 79

4.3 **Instantaneous mean, amplitude, and frequency** of the example signal. Only part of the signal, from $t = 0$ to $t = 0.1$ is shown. The instantaneous frequency, $\omega(t)$, and amplitude, $A(t)$, are presented for the two periodic modes. The mean, $\mu(t)$ of the signal is denoted on the signal itself. 80

4.4 **Decomposition of example signal by NFMD and HHT without noise.** The estimated instantaneous mean mode is shown for both the HHT and NFMD decomposition methods. The instantaneous frequency and amplitude of the two periodic components is also shown for both decomposition approaches. 81

4.5 **Decomposition of example signal with added noise (SNR=35).** The NFMD correctly identifies the instantaneous frequency, amplitude, and mean of the signal, although it does propagate noise in the system. The HHT begins to show erratic behavior in its estimates of all instantaneous parameters. 82

4.6 **Decomposition of signal with discontinuity in instantaneous frequency of a periodic mode.** Data contains additive white noise with SNR=25. The NFMD correctly identifies the three modes, including the instantaneous mean and the frequencies and amplitudes of the periodic modes. The HHT fails to identify the correct number of modes and assigns extra modes to the data. The periodic modes uncovered by HHT do not qualitatively match any of the correct signal modes. 83

4.7 **Decomposition of signal with discontinuity in instantaneous mean.** Data contains additive white noise with SNR=20. The NFMD correctly identifies the periodic mode instantaneous frequency and amplitude, and the instantaneous mean of the signal. The HHT fails to identify either the mean or the periodic component. 84

4.8 **Harmonic oscillator solutions with various forcings.** The left column shows the applied forcing, and the right column shows the solution to the harmonic oscillator assuming it starts from rest. The top row shows a periodic driving force, the middle row shows a non-periodic perturbation, and the bottom row shows a superposition of the two forcings. 85

4.9 **The signal mean (dashed) of the perturbed, periodically driven oscillator matches the solution of the perturbation-only oscillator (orange).** This enables NFMD to directly probe non-oscillatory driving forces applied to oscillators by accurately recovering the signal mean of a nonstationary signal. 86

4.10 **Decomposition of forced oscillator signal by NFMD.** The NFMD decomposes a harmonic oscillator forced with a periodic forcing function and non-periodic perturbation. The instantaneous mean of the signal, $\mu(t)$, is directly correlated to the non-periodic forcing. 87

4.11 **Model fit comparison between instantaneous mean from NFMD and true parameters from non-periodic forcing.** The simulated oscillator is subjected to a perturbation at $t = 0.001$ seconds (left panel). A model is fit to the discovered instantaneous mean mode $\mu(t)$. The time constant τ in the fit model is compared to the time constant of the perturbation force $F_p(t)$ (right panel). 88

4.12 **NFMD decomposition of experimental trEFM control data with controlled perturbation time constants.** A model is fit which contains the time constant τ and compared with truth values. . 89

LIST OF TABLES

Table Number Page

2.1 Trials required for estimating spatial parametric coefficients within 1% error. Error is evaluated in the middle 98% of the problem domain (the interval $x \in [0.1, 9.9]$) with the expression $\|\mathbf{p}_{learned} - \mathbf{p}_{true}\|_2 / \|\mathbf{p}_{true}\|_2$. 27

3.1 Summary of results for three one-dimensional models. The models are provided with the Green's function learned by DeepGreen. A summary box plot shows the relative losses $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_5$, and $\mathcal{L}_6$ for all three model systems. 59

ACKNOWLEDGMENTS

I am extremely grateful for my time working with my advisors, Nathan Kutz and Steve Brunton. Working with them is one of the highlights of my time at University of Washington. They helped me become a better researcher, provided exceptional professional advice, and taught me a significant amount about mathematical physics and machine learning. I have greatly enjoyed every day working with Nathan and Steve; it has made my graduate experience everything I hoped it would be.

I also wish to thank the members of my committee: Fumio Ohuchi, David Ginger, and David Masiello. You have all provided interesting insights into the applications of my works, inspired me to continue this path, and helped me develop new research directions. I especially wish to thank my collaborators. My work with David Ginger and Rajiv Giridharagopal was an excellent collaboration project that taught me about time series analysis and nonlinear oscillators. Furthermore, I must thank Craig Gin for an amazing deep learning collaboration project. Craig guided me through many iterations of the project and taught me a lot about deep learning and the inner workings of neural networks. He was also amazingly helpful for discussions on mathematics and notation.

I would never have made it to this point without the friends I have made throughout my graduate experience. Lily Berger, Alex Peek, Brian Goodall, and Yushi Zang are such good friends to me and helped me navigate the

challenges of graduate school. I am thankful for my friendship with Abbie Ganas, who has kept me motivated through the difficult times and the good times. Craig Bunce, Chris Sandini, and Brandon Rotondo have been three of my closest friends throughout graduate school. I am grateful for our mutual love of science and I cherish their support, which has kept me positive and optimistic.

My wife, Anna, has been my biggest supporter throughout my time in graduate school. She has given me endless advice, listened to me when I faced obstacles, and was always patient with me in the process. I do not think I would have been able to get to this point without her. Additionally, I must thank my parents Jayna and Ed. My parents have supported me from the very beginning to seek higher education and pursue my dreams. Words cannot express my gratitude for having you two as my parents. I would not be the person I am today without you both. I must also thank my siblings Courtney and Dylan. You both supported me and gave me invaluable advice throughout the process.

Finally, I wish to thank my funding sources for making my research possible. Specifically, the UW Molecular Engineering Materials Center (MEM-C), a Materials Research Science and Engineering Center, provided significant financial support throughout my degree through the U. S. National Science Foundation grant DMR-1719797.

DEDICATION

to my wife, Anna, and my parents, Jayna and Ed.

Chapter 1
INTRODUCTION

Data-driven modeling and analysis have guided the development of mathematical physics and advanced our understanding of the universe. In the early 1600s, Johannes Kepler famously took a data-driven approach to fit the planetary orbit of Mars and developed his eponymous laws of planetary motion. Similarly, Isaac Newton developed his laws of motion in the late 1600s using data-driven methods. These laws collectively transformed the worlds of astrophysics and physics, powered by data-driven techniques. As technology has progressed, the types of systems scientists and engineers encounter have become significantly more complicated. Wireless communications, nanoscale mechanics and electronics, interplanetary travel, turbulent fluid dynamics, and nonlinear materials systems all present challenges to traditional data-driven modeling approaches. However, recent developments in machine learning and sensor technology are enabling a new wave of innovation in data-driven modeling. Machine learning has enabled the automated discovery of governing models for physical systems, while sensor technology developments have enabled scientists to take more measurements of their systems than ever previously possible. This dissertation aims to develop data-driven methods for boundary value problems and dynamical systems that are relevant to the world of materials science and engineering.

1.1 Motivation

The developments in this dissertation are communicated for the broader scientific community, although they were all developed with the materials science community

in mind. Specifically, we develop methods for data-driven modeling of boundary value problems and time-frequency analysis of nonstationary time series data. Both BVPs and nonstationary time series signals are ubiquitous in the physical sciences and engineering [1, 2, 3].

In engineering, linear BVPs are used in design applications to describe heat transfer, elasticity and mechanics, and electromagnetics. Linear BVPs have well-developed analytical solution methods including eigenvalue-eigenfunction expansions and the Green's function impulse response of the operator. However, modern systems have given rise to more complex BVPs for systems characterized by nonlinearity and parametric heterogeneities that are not compatible with traditional analytic solutions. These modern systems often require numerical approaches for determining solutions of the system, and traditional data-driven modeling is complicated by the complexity of the governing system model. In this disseration, we present a technique for identifying the governing differential operator in a BVP is presented in Chapter 2 that leverages modern innovations in sparse regression. The method simultaneously allows for identification of a governing nonlinear or linear differential operator *and* its spatially varying parametric coefficients. Furthermore, a technique using deep learning can be employed for linearizing a nonlinear system, as presented in Chapter 3.

Similarly, time-series analysis is critical for a diversity of applications in science and engineering. It has revolutionized the development of test models for observed natural phenomena including planetary motion, chemical reactions, meteorological patterns, and transport phenomena. In a typical scientific workflow, observations are made on a dynamical system and fit to a time series model. Numerous dynamical systems take the form of the humble oscillator, making time-frequency analysis of oscillators, coupled oscillators, nonlinear oscillators, and nonstationary oscillators exceedingly common. However, time series analysis of nonstationary oscillators is nontrivial and numerous developments and contributions have been made to the problem in recent years. Chapter 4 presents a novel data-driven approach to the time-frequency anal-

ysis of nonstationary multimodal time series signals which combines the strengths of modern gradient descent algorithms, the Fourier transform, multi-resolution analysis, and Bayesian spectral analysis. It circumvents many of the shortcomings of classic approaches, enabling the extraction of signal parameters for nonstationary signals with discontinuities in their behavior.

In addition to the broad applicability of both BVPs and time series analysis across the physical sciences and engineering, both have strong applications in materials science and engineering specifically. In materials science, BVPs arise in the study of heat transfer, elasticity, magnetics, electricity, phase transformation, and kinetics. Additionally, time series analysis has many applications including the data-driven modeling of dynamical systems and analysis of oscillators. In this work, the developed methods are applied to mechanical oscillators rather than electronic oscillators although the technique should work equally well on either type. One common use for mechanical oscillators is cantilevers used for nanoscale probes in atomic force microscopy, where deviations from the driving frequency of the oscillator tell a story about the probed system. The techniques and methods developed in this dissertation aim to serve both the broader scientific community, with a focus on materials science and boundary value problems.

1.2 Organization

This dissertation includes three chapters that each introduce a data-driven modeling approach. Chapter 2 describes a method for differential operator identification in boundary value problems which identifies linear and nonlinear operators, and their spatially-varying coefficients, from measurements of forced BVP systems. Chapter 3 presents a deep learning approach for linearizing nonlinear boundary value problems that uses autoencoders to linearize the nonlinear problem and discovers an invertible linear operator in the latent space of the autoencoder. Chapter 4 presents a time series analysis tool for decomposing multimodal nonstationary signals.

1.3 Bibliographic Note

This dissertation is based upon the following publications:

1. Daniel E. Shea, Steven L. Brunton, and J. Nathan Kutz. *SINDy-BVP: Sparse Identification of Nonlinear Dynamics for Boundary Value Problems*. May 21, 2020. arXiv: `2005.10756 [cs]`

2. Craig R. Gin et al. *DeepGreen: Deep Learning of Green's Functions for Nonlinear Boundary Value Problems*. Dec. 31, 2020. arXiv: `2101.07206 [physics]` [1]

3. Daniel E. Shea et al. *Extraction of Instantaneous Frequencies and Amplitudes in Nonstationary Time-Series Data*. Apr. 2, 2021. arXiv: `2104.01293 [cs, eess, math]`

[1] Co-first author

Chapter 2

SINDY-BVP: SPARSE IDENTIFICATION OF NONLINEAR DYNAMICS FOR BOUNDARY VALUE PROBLEMS

In this chapter, we develop a data-driven model discovery and system identification technique for spatially-dependent boundary value problems (BVPs). Specifically, we leverage the *sparse identification of nonlinear dynamics* (SINDy) algorithm and group sparse regression techniques with a set of forcing functions and corresponding state variable measurements to yield a parsimonious model of heterogeneous material systems. The technique models forced systems governed by linear or nonlinear operators of the form $L[u(x)] = f(x)$ on a prescribed domain $x \in [a, b]$.

Section 2.1 provides an overview of the importance of BVPs in the physical sciences, and identifies similar machine learning algorithms for BVPs. Section 2.2 describes the classical SINDy algorithm for dynamical systems, and Section 2.3 develops the modifications made for the SINDy-BVP algorithm and its relationship to the PDE-FIND algorithm. The algorithm is applied to a variety of one-dimensional test problems in Section 2.4 with noisy and clean data, including systems governed by the Sturm-Liouville operator, Poisson's equation, and the biharmonic operator. The results are discussed further in Section 2.5.

2.1 Introduction

Boundary value problems are ubiquitous in the engineering and physical sciences [1, 2]. From heat transfer to elasticity, many fundamental technologies developed in the 20th century are formulated as linear BVPs whose solutions are used in engineering

design. For example, the semi-conductor industry developed many critical technologies and chip architectures by solving BVPs that characterize the underlying quantum, thermal, and electromagnetic physics. Modern BVPs of interest often arise in complex systems characterized by nonlinearity and spatial heterogeneity, thus rendering standard analytic and computational techniques intractable since the governing equations and spatial variability are often unknown. Indeed, the governing BVPs for many emerging applications are often unknown and/or their spatial dependencies undetermined. Modern anisotropic material system design provides a canonical example of the ability to leverage nonlinearity and heterogeneity in order to produce remarkable new materials. Data-driven methods provide a potential theoretical framework for characterizing such materials by discovering both the governing BVPs (linear and nonlinear) and their spatial dependencies through measurements alone. Toward this goal, we develop a sparse regression framework, previously used for the discovery of dynamical systems, in order to discover interpretable and parsimonious BVPs and their spatial dependencies.

The formulation of many canonical problems in physics resulted in the first BVPs. From as early as 1822, when Fourier formulated and solved the heat equation [7], BVPs played a central role in electromagnetism, wave propagation, quantum mechanics, and elasticity. Many of these BVPs resulted from applying a space-time separation of variables decomposition to a governing partial differential equation (PDE). In different geometries and dimensions, the solutions to many canonical BVPs became known as special functions: Bessel, Laguerre, Hermite, Legendre, Chebyshev, spherical harmonics, radial basis, etc. More broadly, these canonical linear equations of mathematical physics were unified under the aegis of Sturm-Liouville theory. The impact of Sturm-Liouville theory in the 20th century is difficult to overestimate given its enormous breadth of applications ranging from the underlying theory of quantum mechanics to the propagation of electromagnetic energy in waveguides. The BVP theory for these two applications arise from a separation of variables solution of the Schrödinger

equation and Maxwell's equations, respectively.

Linear BVPs are amenable to a number of solution strategies, foremost among these being eigenfunction expansions [1]. Such a solution technique is highly advantageous given the interpretability of the eigenfunctions (e.g. quantum mechanical states or propagating waveguide modes) and the many guaranteed mathematical properties of Sturm-Liouville operators, including an orthonormal and complete basis of real eigenfunctions with real eigenvalues for representing solutions. In addition to eigenfunction expansions, there are other methods for generating solutions to BVPs. Most notably is the Green's function [2], which provides an inverse to the Sturm-Liouville operator that can be used to evaluate any forcing of the governing BVP through integration over the so-called fundamental (Green's function) solution. These two traditional and ubiquitous mathematical methods rely on a critical property: linear superposition. Thus any solution can be constructed as a sum of the eigenfunctions appropriately weighted, or the integral (sum) over the fundamental solution. Nonlinear BVPs cannot be handled with such mathematical techniques. Moreover, the spatially varying coefficients of either a linear or nonlinear operator typically requires computational methods to produce solutions. Thus, in many emerging BVPs in the physical and engineering sciences, classical methods which rely on linearity to produce interpretable eigenfunctions or fundamental solutions are ineffective for characterizing the system.

Modern BVPs in science and engineering, which generically take the form $L[u(x)] = f(x)$ with the state variable $u(x)$ and forcing $f(x)$, are typically characterized by the operator L which is nonlinear and highly heterogenous in nature, rendering many of our traditional linear solution strategies ineffectual. This dilemma prohibits development of interpretable solutions, such as Green's functions or eigenfunctions and eigenvalues. Historically, many approaches to this problem have focused on modifying linear models to approximate the nonlinear effects of nonlinear systems. For example, perturbation theory has been used to effectively model weakly nonlinear systems.

Perturbation theory has been applied to a wide variety of nonlinear problems including, among others, nonlinear anisotropic material modeling. These models generally focus on the observed macroscopic system response to applied external stimuli and the agreement between derived theoretical models and experimental data [8, 9, 10, 11, 12]. Entire texts have been written on the subject of anisotropic heterogeneous materials modeling [13], and research in the area remains active.

In this work, we propose a mathematical framework for identifying the governing operator L, including its spatially-varying coefficients, to provide an interpretable understanding of nonlinear and heterogeneous steady-state BVP systems. In many materials systems, spatially-varying parametric coefficients in the operator L are directly tied to the properties of materials in the system. In anisotropic and heterogeneous media, the materials' properties vary with composition and structure, and the mapping between spatial position and local material properties (e.g. heat transfer coefficients, conductivity, diffusivity or porosity) are often not known. Spatially-localized changes in composition and structure can yield significantly different response to external stimuli. Although this work focuses on material science applications, which provide great canonical examples of heterogeneous systems, this mathematical architecture for BVP discovery is fundamentally domain-agnostic and highly flexible.

We propose the SINDy-BVP framework which utilizes data-driven modeling to learn BVP operators directly from data. Our sparse regression framework, which is based upon the *sparse identification of nonlinear dynamics* (SINDy) [14] algorithm, gives rise to interpretable and parsimonious models characterizing the BVP. Our SINDy-BVP framework can identify linear or nonlinear governing equations and/or spatially-varying parametric coefficients of the system from measurement data alone, providing a robust model discovery framework for BVPs. Examples are provided for the Sturm-Liouville operator, a nonlinear modification of the Sturm-Liouville operator, and two illustrative anisotropic, heterogeneous material systems.

In the case of the materials systems, it is important to note that data-driven

modeling represents a paradigm shift in materials modeling. The current standard approach is to model heterogeneous material by their effective macroscopic properties. In contrast, data-driven modeling aims to map the spatially-local material properties.

To our knowledge, there are no other algorithms that focus on identifying the governing operator of a boundary value problem *and* its spatially-varying coefficients simultaneously from data. In [15, 16], a known conservation law or operator is applied to a boundary value problem and a data-driven approach is employed to develop a model where spatially-varying parameters match experimental data set. One approach seeks to find a constitutive relationship between measured variables and applied external forces, but does not identify differential operators thus preventing future numerical solutions from being generated [17]. Other approaches utilize a combination of two separate modules where one module solves the BVP and the other parameterizes the spatially-varying properties of the system [18, 19, 20, 21, 22, 23]. However, one of the two modules is a neural network in all of these two-module approaches, thus preventing discovery of a fully interpretable model. The application of different data-driven modeling approaches for materials modeling has been studied in [24], thus underscoring the importance of this subject.

The paper is outlined as follows: Section 2.2 gives a short background of the SINDy algorithm used extensively in this work. Section 2.3 then formulates the SINDy architecture with boundary value problems for discovery of governing equations and/or their spatially dependent variations. The method developed is applied to a broad range of problems in Section 2.4, including nonlinear boundary value problems. The paper is concluded in Section 2.5 with an overview of the method and a discussion of its outlook on modern nonlinear and heterogenous BVPs.

2.2 Background

This work extends the SINDy family [14, 25, 26] of algorithms to learn the differential operator L in BVPs of the form $L[u(x)] = f(x)$, along with parametric and spatial het-

erogeneous dependencies. SINDy is a model discovery algorithm originally designed to discover governing equations for nonlinear dynamical systems. This method uses a sparse regression framework with a large library of candidate physics models to determine governing equations for physical systems that are often characterized with relatively few terms. This makes the governing equation sparse in the space of possible candidate functions included in the library. SINDy considers dynamical systems of the form:

$$\dot{\mathbf{u}} = \frac{d}{dt}\mathbf{u}(t) = \mathbf{N}(\mathbf{u}, \mathbf{u}^2, ..., \sin(\mathbf{u}), \cos(\mathbf{u}), ...), \tag{2.1}$$

where $\mathbf{u}(t) \in \mathbb{R}^n$ represents the measured variables of the system at time t. The regression is formulated in a discrete matrix formulation where $\mathbf{u}$ is measured at discrete snapshots in time t. The snapshots are used to form the matrices $\mathbf{U}$ and $\dot{\mathbf{U}}$, where $\dot{\mathbf{U}}$ is either directly measured or numerically computed from the snapshots $\mathbf{u}(t)$. If the interval $[0, T]$ is discretized into m points, the two data matrices are the snapshot data matrix $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \dots & u_n(t_1) \\ u_1(t_2) & u_2(t_2) & \dots & u_n(t_2) \\ \vdots & \vdots & & \vdots \\ u_1(t_m) & u_2(t_m) & \dots & u_n(t_m) \end{bmatrix},$$

and the matrix of corresponding time derivatives, $\dot{\mathbf{U}}$:

$$\dot{\mathbf{U}} = \begin{bmatrix} \dot{u}_1(t_1) & \dot{u}_2(t_1) & \dots & \dot{u}_n(t_1) \\ \dot{u}_1(t_2) & \dot{u}_2(t_2) & \dots & \dot{u}_n(t_2) \\ \vdots & \vdots & & \vdots \\ \dot{u}_1(t_m) & \dot{u}_2(t_m) & \dots & \dot{u}_n(t_m) \end{bmatrix}.$$

Since $\mathbf{u}(t)$ is an n-dimensional vector, then the matrices $\mathbf{U}$ and $\dot{\mathbf{U}} \in \mathbb{R}^{m \times n}$. The system identification problem is formulated in matrix form as an over-determined linear regression problem ($\mathbf{Ax} = \mathbf{b}$) for learning the governing equations:

$$\dot{\mathbf{U}} = \boldsymbol{\Theta}(\mathbf{U})\boldsymbol{\Xi}, \tag{2.2}$$

Algorithm 1 SINDy

Input: Candidate functions $\mathbf{\Theta}$, Time derivatives $\mathbf{U}_t$, Regularizer λ, Threshold ϵ, Score function $r(\mathbf{x}) = \|x\|_2$, $iters$

Output: Candidate function loadings $\mathbf{\Xi}$

procedure
  $\mathbf{\Xi} \leftarrow \text{argmin}_{\mathbf{\Xi}'} \|\mathbf{U}_t - \mathbf{\Theta}\mathbf{\Xi}'\|_2$ ▷ Initial $\mathbf{\Xi}$ guess
  for $i = 1, ..., iters$ **do**
    $terms \leftarrow \{i : r(\mathbf{\Xi}(i)) > \epsilon\}$ ▷ Threshold by scored coefficient matrix
    $\mathbf{\Theta} \leftarrow \mathbf{\Theta}[terms]$
    $\mathbf{\Xi} \leftarrow \text{argmin}_{\mathbf{\Xi}'} \|\mathbf{U}_t - \mathbf{\Theta}\mathbf{\Xi}'\|_2 + \lambda\|\mathbf{\Xi}\|_2$ ▷ Repeat regression
  end for
  return $\mathbf{\Xi}$
end procedure

where the matrix $\mathbf{\Theta}(\mathbf{U}) \in \mathbb{R}^{m \times p}$ contains p column vectors, each representing a possible candidate term in the governing equation to be learned. These columns contain candidate symbolic functions for characterizing the governing equations $\mathbf{N}$ in (2.1) by numerically evaluating the state-space at m discrete time points. The unknown coefficient matrix of loadings, $\mathbf{\Xi} \in \mathbb{R}^{p \times n}$, is learned via sparse regression. Candidate model terms in $\mathbf{\Theta}$ can be excluded from the learned governing equation by setting the corresponding coefficient in $\mathbf{\Xi}$ to 0, which is naturally implemented by a sparse regression. The sparse regression minimizes the ℓ_2 reconstruction error (i.e. $\|\dot{\mathbf{U}} - \mathbf{\Theta}\mathbf{\Xi}\|_2$) while enforcing sparsity. Traditional sparse regression uses ℓ_1 (i.e. $|\dot{\mathbf{U}} - \mathbf{\Theta}\mathbf{\Xi}|$) regularization terms, which approximate the computationally challenging non-convex idealized ℓ_0 (i.e. number of non-zero entries in $\mathbf{\Xi}$) regularization. In the SINDy algorithm, sparsity is achieved through an iterative thresholding procedure [14] whose convergence properties have been studied under various assumptions [27, 28]. However, this

problem can be solved using any sparse regression algorithm, such as lasso [29], sparse relaxed regularized regression (SR3) [27, 30], stepwise sparse regression (SSR) [31], or Bayesian methods [32, 33, 34]. The iterative thresholding algorithm for SINDy is outlined in Algorithm 1.

Classical SINDy works well for model discovery and system identification on problems where the terms in the governing equation can be well-represented in the candidate library ($\boldsymbol{\Theta}$) and where the learned terms have constant coefficients with respect to the independent variable(s) of the system (i.e. time-invariant constant coefficients). Parametric PDE-FIND was developed as an extension of the SINDy algorithm to accommodate dynamical systems governed by partial differential equations with time-variant or space-variant coefficients [26]. The PDE-FIND algorithm is modified for the data-driven modeling of BVPs with SINDy-BVP, with a special emphasis on operator identification and parametric coefficient estimation.

2.3 Methods

Our proposed method makes two specific innovations. First, the method learns the differential operator L for BVPs, including spatially varying coefficients. Second, the method is the first application of SINDy to time-invariant systems; all prior works focused on dynamical systems.

These innovations require the method to be adapted for use on BVPs. Previous SINDy methods, including PDE-FIND, were designed for dynamical systems. In dynamical systems, a system can be sampled indefinitely into the future from its initial state. This enables the creation of arbitrarily large data sets, sometimes consisting of thousands or tens of thousands of points. Furthermore, datasets can be easily enriched by sampling systems with different initial states. Neither of these approaches apply for BVP systems.

BVP systems are a more constrained environment for deploying the SINDy algorithm compared to dynamical systems. In BVPs, samples are constrained between

spatial boundaries that specify the domain of the problem, therefore eliminating the possibility of sampling a system's dynamics indefinitely into the future. As previously studied, evaluating a system under a variety of different conditions helps identify the governing model in SINDy and similar algorithms for dynamical systems and is generally superior to increasing the number of samples on a single trajectory generated from one initial condition [35]. This motivates the need to study BVP systems in SINDy-BVP under a variety of different conditions; SINDy-BVP achieves this by applying different forcing functions to the system.

In dynamical systems SINDy, the variable $u = u(t)$ represents the dynamical state variable. In this work, the variable $u = u(x)$ is the state variable for steady-state BVP systems. SINDy-BVP learns the differential operator of a time-invariant system by subjecting the system to a collection of known spatially-varying forcing functions and measuring the system's response. Each response, $u_j(x)$, to a forcing function, $f_j(x)$, is recorded as a trial and each trial is governed by the relationship:

$$\begin{gathered} L[u_j] = f_j \\ j = 1, 2, \cdots, m \\ x \in [a, b], \end{gathered} \tag{2.3}$$

where L is the linear or nonlinear differential operator to be discovered, x is the independent spatial variable, $u_j(x)$ is the measured system state variable quantifying the system's response when subjected to the force $f_j(x)$, and there are m total trials. The $f_j(x)$ are known applied forcing functions, which can be considered as *probes* for the system. The variable x is used to denote a single spatial scalar variable rather than a position vector. The interval $x \in [a, b]$ defines the spatial region of interest, where $x = a$ and $x = b$ are the boundaries of the BVP. Although a variety of boundary conditions can be realized in physical systems, this work uses Dirichlet boundary conditions which specify $u(x = a)$ and $u(x = b)$. The general principle of SINDy-BVP is presented in Figure 2.1 for an operator with two spatially-varying

parameters, $p(x)$ and $q(x)$.

2.3.1 Problem Statement

To begin, we assume L is a second-order differential operator. This operator order assumption will be relaxed in later sections (Section 2.4.4). If L is second order, it is known that $L[u]$ contains the term u_{xx}. If u_{xx} is in the governing equation $L[u(x)] = f(x)$, we assume it can be represented as some generalized function N which contains $f(x)$ and other terms in L:

$$u_{xx} = N(u, u^2, u^3, ..., u_x, ..., f(x)). \tag{2.4}$$

This is the BVP equivalent to (2.1). The BVP problem (2.4), which is formulated as a continuous variable over the domain $x \in [a, b]$, is discritized into n spatial locations. We assume these to be equally spaced measurements or discretization locations. The discretized function $u(x)$ is mapped to the vector $\mathbf{u} = [u(x_1)\ u(x_2)\ u(x_3)\ \cdots\ u(x_n)]^T$ where $x_1 = a$ and $x_n = b$. With the vectorization of the data, we can adopt the SINDy nomenclature and restate the sparse regression for BVPs as

$$\mathbf{U}_{xx} = \mathbf{\Theta}(\mathbf{U}, \mathbf{F})\mathbf{\Xi}, \tag{2.5}$$

where $\mathbf{U}_{xx}$ is the second spatial derivative of the discretized vector of the state space $u(x)$, $\mathbf{\Theta}$ is a library of candidate basis functions believed to comprise $N(\cdot)$, and $\mathbf{\Xi}$ is a vector of coefficients which prescribe the loadings of the columns of $\mathbf{\Theta}$. The coefficient vector can vary spatially with x, or more precisely, the discretization of x.

This regression uses input data consisting of matrices $\mathbf{U} \in \mathbb{R}^{m \times n}$ and $\mathbf{F} \in \mathbb{R}^{m\ timesn}$ with n discrete sampled spatial positions and m unique trials or forcings. Each trial is a system response $\mathbf{u}_j$ to a corresponding forcing function $\mathbf{f}_j$ governed by the same

operator L. The input data set $\mathbf{U}$ and $\mathbf{F}$ have the structure:

$$\mathbf{U} = \begin{bmatrix} u_1(x_1) & u_1(x_2) & \dots & u_1(x_n) \\ u_2(x_1) & u_2(x_2) & \dots & u_2(x_n) \\ \vdots & \vdots & & \vdots \\ u_m(x_1) & u_m(x_2) & \dots & u_m(x_n) \end{bmatrix} \tag{2.6}$$

$$\mathbf{F} = \begin{bmatrix} f_1(x_1) & f_1(x_2) & \dots & f_1(x_n) \\ f_2(x_1) & f_2(x_2) & \dots & f_2(x_n) \\ \vdots & \vdots & & \vdots \\ f_m(x_1) & f_m(x_2) & \dots & f_m(x_n) \end{bmatrix}. \tag{2.7}$$

Note this is different from dynamical systems SINDy, where m temporal snapshots of a dynamical system are sampled and the state vector $\mathbf{u}(t)$ has n components. In SINDy-BVP, the $\mathbf{U}$ samples the spatial positions $x_1, x_2, \cdots, x_n$ for m different trials, where each trial is forced by a different forcing function. Additionally, the outcome variable (or left-hand side) in this formulation is a spatial derivative of $\mathbf{U}$, *not* the typical $\mathbf{U}_t$ seen in dynamical systems SINDy formulations. The spatial derivatives of $\mathbf{U}$ are generated by numerical differentiation to produce $\mathbf{U}_x$, $\mathbf{U}_{xx}$, and higher order derivatives as needed. The numerically differentiated data is stacked as a vector and used as the outcome variable for the SINDy regression (2.5). The stacked vector $\mathbf{U}_{xx}$

has the structure:

$$\mathbf{U}_{xx} = \begin{bmatrix} u_{1xx}(x_1) \\ u_{2xx}(x_1) \\ \vdots \\ u_{mxx}(x_1) \\ u_{1xx}(x_2) \\ u_{2xx}(x_2) \\ \vdots \\ u_{mxx}(x_2) \\ u_{1xx}(x_n) \\ u_{2xx}(x_n) \\ \vdots \\ u_{mxx}(x_n) \end{bmatrix}. \tag{2.8}$$

The candidate function matrix $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F}) \in \mathbb{R}^{(m\times n)\times(p\times n)}$ is constructed as a sparse block matrix to enable discovery of spatially-varying parametric coefficients. The library is constructed to allow a separate coefficient to be discovered for each spatial position, x_k. $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ contains columns for each of the p symbolic candidate basis functions, each evaluated at the n spatial coordinates for all of the m trials. The candidate model functions included in the library $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ are further described in Section 2.3.5. The matrix $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ is a diagonal sparse matrix with the structure:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}^{(1)} & & & & \\ & \ddots & & & \\ & & \boldsymbol{\Theta}^{(k)} & & \\ & & & \ddots & \\ & & & & \boldsymbol{\Theta}^{(n)} \end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{\Theta}^{(k)} \in \mathbb{R}^{m\times p}$ is a symbolic function library with p candidate function columns

evaluated for each of the m trials at spatial coordinates, x_k:

$$\boldsymbol{\Theta}^{(k)} = \begin{bmatrix} u_{1,k} & \dots & (u_{1,k})_x & \dots & (u_{1,k})^2 & \dots & f_{1,k} \\ & & \vdots & & & & \\ u_{j,k} & \dots & (u_{j,k})_x & \dots & (u_{j,k})^2 & \dots & f_{j,k} \\ & & \vdots & & & & \\ u_{m,k} & \dots & (u_{m,k})_x & \dots & (u_{m,k})^2 & \dots & f_{m,k} \end{bmatrix},$$

where the subscripts j and k refer to the trial number and spatial coordinate, respectively. The library $\boldsymbol{\Theta}^{(k)}$ allows discovery of the parametric coefficients at spatial position x_k. This construction requires different forcings for each $\boldsymbol{\Theta}^{(k)}$ so that the regression 2.5 is not underdetermined and lacking insufficient constraints. The forcing functions $f_j(x)$ are included in the library because they are known to influence the observed behavior of the system. Further, as described in Section 2.3.4, they must be in the learned function $N(\cdot)$, and therefore must be included in the candidate term library $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ to learn an accurate operator.

The problem is formulated as a group regression problem. Candidate model functions are tied together with a set of group indices G. G is a set of tuples of indices, where there are p tuples in the set G and each tuple contains n indices. Each tuple identifies related rows in $\boldsymbol{\Xi}$ and columns in $\boldsymbol{\Theta}$ that correspond to the same candidate function. For example, consider the candidate function $u_{1,k}$ in the library $\boldsymbol{\Theta}^{(k)}$. It is the candidate function in the first of p columns in $\boldsymbol{\Theta}^{(k)}$. There is a tuple of indices that can be constructed which refers to the first column of every matrix in equation (2.9) (i.e. $\boldsymbol{\Theta}^{(1)}, \boldsymbol{\Theta}^{(k)}, ..., \boldsymbol{\Theta}^{(n)}$). This example tuple would contain values $(1, (p+1), (2p+1), ..., (n-1)p+1)$. The tuples in set G can be generated by the relationship $G = \{g_l = l + (p \times k) : k = 1, \dots, n; l = 1, \dots, p\}$ where l counts through the p candidate functions and k counts through the n discrete spatial positions. Each tuple $g_l \in G$ contains column indices of $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ and row indices of $\boldsymbol{\Xi}$ corresponding to a single candidate function at all of the n discrete spatial positions.

Group sparsity is imposed to produce a solution which is sparse in the space of

possible candidate functions and where the coefficient can vary with spatial position x. Group sparse regression is performed using the *Sequential Grouped Threshold Ridge Regression* (SGTR) algorithm developed by Rudy [26]. An intuitive way of thinking about this approach is presented in Fig. 2.2. In the figure, a separate sparse regression is constructed for each of the n spatial coordinates. The regression aggregates data from m trials and enforces the solution's sparsity pattern across all of the spatial positions. As implied by the figure, this allows for inference of the operator L and its parametric coefficients.

2.3.2 Sequential Grouped Threshold Ridge Regression

SGTR [26] is a group regression technique which accomplishes group-level sparsity through an iterative thresholding process. This example assumes SINDy-BVP will be performed with the outcome variable $\mathbf{U}_{xx}$. Using the construction provided above, each group in G contains a set of indices which represent columns of a single candidate term in $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ at all spatial positions and its corresponding coefficient in $\boldsymbol{\Xi}$ at all spatial positions. The algorithm, shown in Algorithm 2, achieves sparsity at the group level through a combination of ridge regression and iterative thresholding across all groups.

The iterative thresholding loop in the SGTR algorithm progressively eliminates groups from $\boldsymbol{\Xi}$ and $\boldsymbol{\Theta}$ by setting the columns in $\boldsymbol{\Theta}$ to zero. This thresholding imposes sparsity on the candidate functions in $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ based on the candidate function's coefficient vector $\boldsymbol{\Xi}$. The evaluation function r in this work is the ℓ_2 norm, which means SGTR performs ridge regression and thresholds out candidate functions based on the ℓ_2 norm of the coefficient vector (i.e. $r(\boldsymbol{\Xi}^{(g)}) = \|\boldsymbol{\Xi}^{(g_l)}\|_2$). The result is a parsimonious function for $\mathbf{U}_{xx}$ where the non-zero coefficients are allowed to vary at each spatial position x_k.

2.3.3 Model Selection

The optimal model is selected from a set of candidate models generated by varying the thresholding value ϵ in the SGTR algorithm. A range of tolerance values ϵ are computed by:

$$\begin{aligned}\epsilon_{max} &= \max_{g\in G} \|\Xi^{(g)}_{ridge}\|_2 \\ \epsilon_{min} &= \min_{g\in G} \|\Xi^{(g)}_{ridge}\|_2 \\ \Xi_{ridge} &= (\boldsymbol{\Theta}(\mathbf{U},\mathbf{F})^T\boldsymbol{\Theta}(\mathbf{U},\mathbf{F}) + \lambda I)^{-1}\boldsymbol{\Theta}(\mathbf{U},\mathbf{F})^T\mathbf{U_{xx}},\end{aligned}$$

where ϵ_{max} and ϵ_{min} are the highest and lowest tolerances that affect the sparsity of the predicted model, and λ is a regularization constant. The ridge regression regularization constant is held constant at $\lambda = 10^{-5}$ for all problems in this work. At a thresholding value of ϵ_{max}, all coefficients are set to 0 after the first thresholding step with SGTR. Conversely, using ϵ_{min} as the thresholding tolerance would not eliminate any candidate functions with SGTR. A number of values, typically 50, spaced logarithmically between ϵ_{max} and ϵ_{max} are used to compute the candidate models.

The optimal model is then selected from the candidate models by choosing the model which minimizes the PDE-FIND loss function [26]:

$$\mathcal{L} = N\ \ln\left(\frac{\|\boldsymbol{\Theta}(\mathbf{U},\mathbf{F})\boldsymbol{\Xi} - \mathbf{U}_{xx}\|_2^2}{N} + \beta\right) + 2k, \tag{2.10}$$

where k is the number of nonzero coefficients in the identified model ($k := \|\boldsymbol{\Xi}\|_0/m$), β is a small constant, and $N := m \times n$ is the number of rows in $\boldsymbol{\Theta}(\mathbf{U},\mathbf{F})$. The loss function used to select the model assumes there is error in numerical differentiation used to compute $\mathbf{U}_{xx}$, and thus a model that minimizes only mean squared error ($\|\boldsymbol{\Theta}(\mathbf{U},\mathbf{F})\boldsymbol{\Xi} - \mathbf{U}_{xx}\|_2^2$) is likely overfit. Overfit models are balanced by the parameter β, which allows for some misfit and simultaneously prevents the occurrence of $\ln(0)$ in the loss function. The constant β is fixed in this work as $\beta = 10^{-6}$.

Similar to previous SINDy works, the algorithm exhibits improved performance when each candidate function is normalized to unit length [14]. When constructing the block diagonal matrix $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$, the entries $\boldsymbol{\Theta}^{(k)}$ are stacked and each column is normalized to unit length. More precisely, the matrix $\hat{\boldsymbol{\Theta}} \in \mathbb{R}^{(m \times n)xp}$ which is assembled as $\hat{\boldsymbol{\Theta}}^T = [\boldsymbol{\Theta}^{(1)}, ..., \boldsymbol{\Theta}^{(k)}, ..., \boldsymbol{\Theta}^{(n)}]$. $\hat{\boldsymbol{\Theta}}$ is normalized column-wise over each of the p columns, each containing a candidate function. Similarly, the outcome variable vector (e.g. $\mathbf{U}_{xx}$) is normalized such that $\|\mathbf{U}_{xx}\| = 1$.

2.3.4 Learning the Operator L

In the previous sections, a method was described for learning a function that describes $u_{xx}(x)$, but without connecting that function to the operator. In this section, we will show how the operator L can be inferred from the learned function for $u_{xx}(x)$. Using a simple ansatz that the differential operator is at least second order, it is presumed the operator contains a $u_{xx}(x)$ term. This means we can define a new function $\mathbf{N}$ such that:

$$\begin{aligned} & N = Lu(x) - \phi(x)u_{xx}(x) \\ \implies \quad & Lu(x) = N + \phi(x)u_{xx}(x). \end{aligned}$$

If $u_{xx}(x)$ has the spatially-varying coefficient $\phi(x)$, then the function N is related to $\phi(x)$ and the operator L by rearrangement of the original problem to:

$$u_{xx}(x) = \frac{1}{\phi(x)}(f(x) - N). \tag{2.11}$$

This formulation shows the parametric coefficient for the term $f(x)$ is $1/\phi(x)$. Because $u_{xx}(x)$ and the forcing function $f(x)$ are known, and the loadings on the forcing function $(1/\phi(x))$ are discovered by SINDy-BVP, this relationship allows us to determine the governing operator L. Note that this specific formulation prohibits discovery of conservation laws that take the form $Lu = 0$, as a forcing is required to discover the loading $\phi(x)$ and there would be insufficient information in the regression (2.5)

to discover spatially-varying coefficients in the operator L. The terms with nonzero $\boldsymbol{\Xi}^{(g)}$, other than $u_{xx}(x)$, correspond to the function N, and represent additional terms in the operator L. Each of these coefficients is learned as a vector $\boldsymbol{\Xi}^{(g)} \in \mathbb{R}^n$. By identifying $\phi(x)$, we can directly infer the operator L from the learned function for $u_{xx}(x)$, $f(x)$, $\phi(x)$, and N. This method can be extended to differential operators of any order and form, including fourth order linear operators and nonlinear operators.

There is a simpler formulation with $f(x)$ as the outcome variable (or left-hand side term) in the regression formulation, which theoretically provides the opportunity to directly learn the operator L through a regression of the form $\mathbf{F} = \boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})\boldsymbol{\Xi}$. However, including a numerically accurate $\mathbf{F}$ in the library $\boldsymbol{\Theta}$ improves the ability of SINDy-BVP to handle noise while identifying the operator and its parameters. If $\mathbf{F}$ also contained significant noise, there may not be an advantage to this construction.

2.3.5 Candidate Function Library

The candidate function library, $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$, contains columns for derivatives of $u(x)$, nonlinearities of $u(x)$, and forcing functions $f(x)$. In all cases, $\boldsymbol{\Theta}$ contains $u(x)$, and polynomials of $u(x)$ up to fifth order.

The derivatives in the library depend on the outcome variable. Assume the outcome variable for SINDy-BVP is $\mathbf{U}_{xxxx}$, the discrete form of $d^A u(x)/dx^A j$. In this case, $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{F})$ contains derivatives $d^a u(x)/dx^a$ of order a, for integers $0 < a < A$. For example, if $\mathbf{U}_{xxxx}$ is the outcome variable, the library contains columns for the derivatives $u_x(x)$, $u_{xx}(x)$, and $u_{xxx}(x)$. Furthermore, the products of $u(x)$ and nonlinearities in $u(x)$ with the spatial derivatives of $u(x)$ are included in $\boldsymbol{\Theta}$ (e.g.uu_x and $u^2 u_{xx}$). Finally, a column for the forcing functions is included in $\boldsymbol{\Theta}$ containing all data in $\mathbf{F}$.

In general, the constructed candidate basis function library must include the basis functions in the governing model. If the terms contained in the operator are *not* present in the library, the learned governing operator will be inaccurate and/or

incomplete. These failure modes have been previously discussed in [14].

2.4 Computational Results

2.4.1 Boundary Value Problem Models

The models used in this work are solved on the interval $x \in [0, 10]$ using the shooting method [36] with 1000 grid points. A tolerance of 0.001 is used for the right-side boundary condition, such that solutions which aim to achieve $u(x = 10) = 0$ can have an actual value $u(x = 10) \in [-0.001, 0.001]$. The following subsections describe the models used for this work.

Linear Sturm-Liouville

Sturm-Liouville form operators are an extremely common class of linear, self-adjoint, Hermitian operators. Sturm-Liouville theory is especially important in engineering applications, and its study focuses on operators of the form in Equation 2.12:

$$L[u] = [-pu_x]_x + qu \qquad x \in [0, 10], \tag{2.12}$$

where the state variable $u(x)$ is a function of the spatial variable x, and $p(x)$ and $q(x)$ are in general functions of the spatial variable. In our example model, the parametric coefficients are described by the functions:

$$p(x) = 0.5\ \sin(x) + 0.1\ \sin(12x) + 0.25\ \cos(4x) + 2$$
$$q(x) = 0.4\ \sin(3x) + 0.15\ \cos(8x) + 1.$$

The boundary conditions $u(0) = 0$ and $u(10) = 0$ are enforced for solutions of this model. The parametric coefficients in this model, $p(x)$ and $q(x)$, were selected to provide an example to demonstrate SINDy-BVP on a linear model with rapidly-changing spatially-varying coefficients.

Nonlinear Sturm-Liouville

A quadratic nonlinearity can be introduced to the Sturm-Liouville model in the following form:

$$L[u] = [-pu_x]_x + qu + \alpha q u^2 \qquad x \in [0, 10], \tag{2.13}$$

where α controls the extent of nonlinearity in the term $\alpha q u^2$. The value $\alpha = 0.4$ is used. The parametric coefficients $p(x)$ and $q(x)$ are described by:

$$p(x) = 0.5\sin(x) + 0.1\sin(11x) + 0.25\cos(4x) + 3$$
$$q(x) = 0.6\sin(x+1) + 0.3\sin(2.5x) + 0.2\cos(5x) + 1.5.$$

Boundary conditions $u(0) = 0$ and $u(10) = 0$ are used for this model. Again, the parametric coefficients in this model, $p(x)$ and $q(x)$, were selected to demonstrate SINDy-BVP on a nonlinear model with rapidly changing spatially-varying coefficients.

Linear Second Order Poisson

Many simple physical systems are described by Poisson's equation. These elliptic differential equations are described by a Laplacian operator subjected to a force: $\Delta u = f$. In our system, a parametric coefficient describing a material property, $p(x)$, is introduced to the model.

$$L[u] = [-pu_x]_x \qquad x \in [0, 10]. \tag{2.14}$$

Steady-state heat conduction is one example of a system that follows from this model. The coefficient $p(x)$ could thus be considered as thermal diffusivity (often κ) of the material and is allowed to vary spatially. The material in this example system is a two-component composite that is anisotropic along the x coordinate and contains an exponentially-varying quantity of the two materials along the x direction. The model for $p(x)$ in this problem is the simple arithmetic average:

$$p(x) = v_a(x)p_a + v_b(x)p_b,$$

where $v_a(x)$ and $v_b(x)$ are the volume fractions of component a and b respectively, and vary spatially. The values p_a and p_b are the material properties for pure a and b. The components' material properties hold the value $p_a = 12$ and $p_b = 3$, which do not change.

Although this model is simple and the arithmetic average often overestimates the true observed material properties of composites [13], it is instructive to consider the ability of SINDy-BVP to learn an operator for a system with a spatially varying anisotropic material property. The volume fraction of component b is described by an exponential decay function while component a makes up the remainder of the volume:

$$v_a(x) = 1 - v_b(x)$$
$$v_b(x) = 0.1 - 0.7\exp(0.4x)$$

A steady-state heat conduction problem, where one end has a higher temperature than the other, is modeled in this problem. Boundary conditions of $u(0) = 0.8$ and $u(10) = 0$ are applied.

Euler-Bernoulli Beam Theory

The Euler-Bernoulli beam theory uses the biharmonic fourth order linear operator from elasticity theory to describe beam deflections given mechanical properties of the beam. The operator takes the form:

$$L[u] = [-EIu_{xx}]_{xx} \qquad x \in [0, 10], \tag{2.15}$$

where EI is the flexural rigidity of the material. In our model, the flexural rigidity varies spatially following a stepwise function as expected for a lamellar, laminate

composite with the lamella oriented perpendicular to the x coordinate:

$$EI(x) = \begin{cases} 10 & 0 \leq x < 2, \\ 2.5 & 2 \leq x < 4, \\ 10 & 4 \leq x < 6, \\ 5 & 6 \leq x < 8, \\ 2.5 & 8 \leq x \leq 10 \end{cases} .$$

The stepwise function $EI(x)$ is a challenge for SINDy-BVP because of the discontinuities at the jumps in flexural rigidity which occur at $x = 2$, $x = 4$, $x = 6$, and $x = 8$. The beam in this problem is considered clamped at both ends such that $u(0) = 0$ and $u(10) = 0$.

Forcing Functions

The forcing functions for all examples are sinusoidal functions of the form $a\ \sin(bx)+c$. The amplitude a, frequency b, and positive offset c are selected from a set of values which varies for each model. This family of functions was selected to enable rapid generation of a large data set with solutions of similar order of magnitude. The parameters a, b, and c are selected for each system to produce solutions $u(x)$ where $\|\mathbf{u}\|_\infty \approx 1$.

The approach for this work is to generate a large library of solutions to the problem $L[u_j(x)] = f_j(x)$, and then randomly sub-sample the library of solutions to test the SINDy-BVP algorithm. Prior works suggest that a variety of different system conditions must be tested to optimize discovery of the underlying governing equations of a system [35]. A weak formulation of SINDy also suggests that the optimal test functions to use for a system would have high values of their derivatives at points where the highest error in the model exists [37]. However, knowledge of the region with the highest model error would not be known prior to collecting data for a physical

test system, as it would require fitting and testing a model. For this reason, we choose to randomly sample a database of randomly generated functions producing solutions of similar magnitude.

This approach is physically and experimentally relevant. In real systems, a number of different conditions could be tested and compiled into a database of forcings $\mathbf{F}$ and corresponding responses $\mathbf{U}$ on the discretized spatial vector $\mathbf{x}$.

2.4.2 Operator Identification and Parametric Coefficient Estimation

SINDy-BVP aims to achieve two primary goals: identification of the structure of a differential operator L and discovery of the parametric coefficients present in L for a forced system governed by the model $L[u(x)] = f(x)$. The method is applied to the four models described in Section 2.4.1. Operator identification is only required in cases where the governing operator is unknown, and so two cases can be considered: known operator and unknown operator. The data used in this section is noise-free (up to numerical precision). Derivatives are computed using the finite differences method. Although this is physically unrealistic since measurements would introduce noise or rounding errors, this exercise provides insight into the capability of the method.

Fig. 2.3 shows the four models used in this paper, three example trials used for training the SINDy-BVP models, and a plot of the parametric coefficients learned by SINDy-BVP compared to the true parameters in the operator L over the interval $x \in [0, 10]$. The parametric coefficient plots are taken from the case of an "unknown operator", where both the operator and the parametric coefficients are learned by SINDy-BVP.

SINDy-BVP is effective at learning the coefficients $p(x)$ and (if applicable) $q(x)$ with relatively few trials for numerical precision data. Table 2.1 shows the number of trials required for SINDy-BVP to estimate the parametric coefficients to within 1% error for the middle 98% of the interval (i.e.[0.1,9.9]). This metric is used to quantify the accuracy of learned coefficients because the error in the learned coefficients

Table 2.1: Trials required for estimating spatial parametric coefficients within 1% error. Error is evaluated in the middle 98% of the problem domain (the interval $x \in [0.1, 9.9]$) with the expression $\|\mathbf{p}_{learned} - \mathbf{p}_{true}\|_2 / \|\mathbf{p}_{true}\|_2$.

Trials Required	Known L	Unknown L
Linear Sturm-Liouville	6	25
Nonlinear Sturm-Liouville	6	10
Linear Second Order Poisson	2	8
Euler-Bernoulli Beam Theory[1]	4	4

happens almost exclusively at the boundaries (inspect Fig. 2.3).

2.4.3 Effects of Noise

The effect of noise is studied separately for the problems of operator identification and parameter estimation. Noise is introduced to each system by applying Gaussian white noise to the measurement data in $\mathbf{U}$ over each row, $\mathbf{U}_j$. The noise is defined by a signal-to-noise ratio (SNR) using the relationship $\text{SNR} = 10 \log_{10}(\|u(t)\|_2^2 / \|(\tilde{u}(t) - u(t)\|_2^2)$. In order to enable differentiation of noisy input data, numerical differentiation is performed using a windowed Chebychev polynomial interpolation method. In the windowed interpolation method, a subset of 20 continuous data points is selected are fit to a fifth order Chebychev polynomial. In the last example with a fourth-order derivative, 30 points are used in the window with a sixth order polynomial. Derivatives of the fit polynomial function are used as derivative data for the regression.

Noisy Operator Identification

Operator identification is a challenging task for SINDy-BVP with noise. Prior works with SINDy have also described challenges in dealing with noise, so this is not a surprising finding. Fig. 2.4(a) shows the effect of varying the SNR ratio, and 2.4(b)

shows the effect of increasing the number of trials used in regression at a constant SNR of 100. The phrase "spurious terms" in Fig, 2.4 refers to the number of incorrectly identified monomial basis functions in the operator L, including both erroneous terms (terms that do not exist in the operator) and missing terms (terms that should be in the operator, but are not identified by the algorithm). The "erroneous terms" category is the most common error, where the SGTR algorithm fails to find a parsimonious model describing the system and includes additional terms to approximate the noisy training data accurately.

The Sturm-Liouville model shown in the top row of Figure 2.4 show that the operator identification task succeeds with an SNR=100, but begins to fail at lower SNR. Furthermore, the operator identification task succeeds with as little as 10 trials at SNR=100, showing that SINDy-BVP is robust at high SNR. In the case of the Nonlinear Sturm-Liouville model, over 180 trials are required to routinely identify the correct model at SNR=100. With a constant 200 trials used, spurious terms show up in the learned function starting below the threshold of SNR=100. Operator identification results are shown for the Poisson model in the third row. Similar to the Linear and Nonlinear Sturm-Liouville models, the Poisson model requires at minimum a SNR of 100. With an SNR of 100, the operator identification task succeeds for nearly any number of trials. However, a curious peak happens at 80 and 90 trials where an extra term is identified in the operator (specifically 'u' is added to the model). Despite the added term, the change in the loss value is relatively low, indicating that the added term makes a small difference in the accuracy of the model. Although SINDy-BVP identifies the extra term, the identified model is still relatively sparse and would provide an excellent starting point for parameter estimation. The Euler-Bernoulli beam, shown in the last row, is prohibitively challenging to identify with noise in the signal. Although signals with SNR over 200 can successfully identify the beam model, operator identification fails with any number of trials at SNR of 100.

One common error in model convergence is the inclusion of candidate model terms

with a few 'large' values in its coefficient vector $\boldsymbol{\Xi}^{(g)}$. The large values pass the thresholding step, which is based on the ℓ_2 norm of the entire coefficient vector. This error effectively includes candidate terms in the final model that have relatively small influence on the model predictions, but that satisfy errors from noise in the training data. This error could potentially be mitigated by enforcing ℓ_∞ constraints on the coefficient vectors, or by enforcing local smoothness of the learned coefficient.

In contrast, a term is occasionally excluded from a learned model. For example, let the differential operator be the linear Sturm-Liouville form $L[u] = -p(x)u_{xx} - p_x(x)u_x + q(x)u = f$. If the training data exhibits relatively little contribution from the term $-p_x(x)u_x$, SINDy-BVP may exclude the u_x candidate term from the learned model. This results in an inaccurate model of $L[u] = -p(x)u_{xx} + q(x)u$. Although this model is missing the u_x term, it has relatively low loss function values and may be selected as the correct model.

These two errors account for the most common forms of error at low noise levels. At higher noise levels ($> 5\%$ noise) the regression begins to add terms to the learned model to accomodate noise in the measurements, which is a common form of error in SINDy and other sparse regression algorithms.

Noisy Parameter Estimation

If the operator is known, the focus shifts towards estimating the spatially-dependent parametric coefficients. Fig. 2.5 shows the effect of noise on parameter estimation by computing a "Coefficient Error". The coefficient error is defined as $E_p = \|\hat{\mathbf{p}} - \mathbf{p}\|_2/\|\mathbf{p}\|_2$, where $\hat{\mathbf{p}}$ is the predicted coefficient vector and $\mathbf{p}$ is the true coefficient vector. The task is quantified in a similar way to the operator identification task, where the SNR of input data and number of trials used for regression are varied. In Fig. 2.5(a), 200 trials are used as input data and the SNR is modulated between 10 and 1000. In Fig. 2.5(b), the SNR is fixed to 100 and the number of trials is varied between 10 and 200. For visual clarity in the plots, the coefficient error is capped at

1, so that $min(1, E_p)$ is plotted.

The parameter estimation task is successful for all four models at SNR greater than 200. Parameters for the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, and Poisson models can also be estimated at SNR of 100, and the Sturm-Liouville models are somewhat successful at SNR=50. With a fixed SNR=100, the parameters for the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, and Poisson models can be accurately identified within about 5% for any number of trials over 10 trials. However, the parameters for the Euler-Bernoulli Beam equation cannot be accurately identified with even 200 trials at SNR of 100. Similar to the operator identification task, SINDy-BVP appears to succeed in most cases with SNR greater than 100. These results indicate that collection of clean data is the most important aspect of using SINDy-BVP for operator identification in experimental systems. Hyperparameter tuning and data filtering may improve these results, but aiming for an SNR over 100 is a critical step for practical use of SINDy-BVP.

2.4.4 Model Differential Order Selection

This section addresses the need to identify the derivative order of the model's left hand side. In the Euler-Bernoulli beam theory example, for instance, the outcome variable should be $\mathbf{U}_{xxxx}$. This example will be used to show that a set of test trials can be used to determine the best model for a collection of different outcome variables.

Using the methods described in section 2.3.4, the operator L can be identified from a generalized equation N which describes a given left-hand side term. A series of SINDy-BVP regressions is used to identify a model operator L for each outcome variable in a set with increasing differential order ($\mathbf{U}_x$, $\mathbf{U}_{xx}$, $\mathbf{U}_{xxx}$, $\mathbf{U}_{xxxx}$, etc.). Each of these operators is then evaluated with the test trials for the error $\mathcal{L}_{test} = 1/T \sum_{j=1,\dots,T} \|L[\mathbf{U}_j] - \mathbf{F}_j\|$ for the T test trials.

Fig. 2.6 shows how $\mathcal{L}_{test}$ compares between data-driven models generated with different outcome variables. The model which minimizes the test error is the model

for $\mathbf{U}_{xxxx}$, indicating this is the correct model to use. This approach emphasizes the governing relationship, $Lu = f$. In this example, the test data set contains 45 trials.

2.5 Discussion

SINDy-BVP successfully extends the data-driven modeling approach of SINDy from dynamical systems to time-invariant, steady-state, spatially-varying BVP systems. The method is used to identify a differential operator, L, governing forced systems of the form $L[u_j(x)] = f_j(x)$, where $f_j(x)$ is a known forcing function and $u_j(x)$ is the measured variable which quantifies the system's response to the forcing. The operator L can be nonlinear or linear.

Operator identification and parametric coefficient estimation are the two most important tasks. With numerical precision data, SINDy-BVP is effective at identifying the operator and the parametric coefficients within 1% error with relatively little data (see Table 2.1). However, noisy data makes both tasks more difficult. Fig. 2.4 indicates operator identification is challenging with as little as 1% noise. Parameter estimation can succeed within 15% error with as little as 10-15 trials ($f_j(x)$-$u_j(x)$ pairs) in 1% noise (Fig. 2.5(b)). However, parameter estimation error does not improve significantly unless much larger sets of training data (over 100 trials) are used. With and without noise in the data, SINDy-BVP often incurs error in both operator identification and parameter estimation near the boundaries of the system. The boundary error is likely a result of the ill-conditioned inverse problem near the boundaries. The problem is ill-conditioned because the boundary conditions are identical in all trials. Specifically, the Dirichlet boundary conditions used in this work stipulate the boundary values $u_j(a)$ and $u_j(b)$ are the same in all trials.

There are three primary challenges facing SINDy-BVP. First, model identification relies on having the correct terms in the regression basis function library. If the library is missing one of the basis functions in the operator, the regression will attempt to approximate the missing term using remaining basis functions in the li-

brary [14]. The difference between an incomplete learned operator and true operator is beyond the scope of the present work. The second challenge for SINDy-BVP is its susceptibility to noise. Noise can result in both extra added terms and missing excluded terms, both contributing error to the learned model. Noise is often amplified by numerical differentiation methods, so one promising approach to reducing noise is integral-based formulations of SINDy which were shown to improve noise-handling[38]. Alternatively, improved differentiation methods could be developed or black-box interpolation methods (e.g. neural networks) could be used to build 'clean' signals $\mathbf{U}_j$ from noisy data. Finally, judicious selection of training data is critically important. This is true for any data-driven modeling approach. In the case of SINDy-BVP, the data must exhibit relatively equal contribution to the system behavior from each of the terms in the governing operator for the algorithm to learn the complete and correct operator.

The block matrix regression in Equation 2.9 empowers SINDy-BVP to learn parametric coefficients described by nontrivial functions including multimodal sinusoidal and piecewise functions. However, it also imparts minor drawbacks. The discovered parametric coefficients are learned as a vector $\boldsymbol{\Xi}^{(g)} \in \mathbb{R}^n$, where each value of the coefficient is mapped to a measured spatial position x_k. This explicitly ties the resolution of the learned parameters to the measurement grid. If it is reasonable to expect the parametric coefficients to be described by a set of basis functions, it is plausible to use the learned coefficient vectors as part of a sparse symbolic regression problem akin to SINDy where the coefficients are described by a sparse combination of basis functions. Although this approach could reduce the number of measurement points required to learn the parametric coefficients, it implicitly depends on projecting the functions describing the coefficients into a (known) sparse function basis. If SINDy-BVP was applied to higher-dimensional systems (2D or 3D), the regression (2.9) grows exponentially with each dimension. For example, a 2D model on a grid with n samples in each dimension requires a regression 2.9 with n^2 matrices $\boldsymbol{\Theta}^{(k)}$

comprising the block diagonal matrix $\boldsymbol{\Theta}$. However, a recent work shows promising results on handling higher-dimensional systems with tensor-based SINDy methods [39]. Applying SINDy-BVP to higher-dimensional systems is a subject for future work.

A variety of other adaptations to the SINDy-BVP architecture could also be made which may improve model convergence for operator identification and the accuracy of parametric coefficient estimation. Noise handling may be improved by implementing ℓ_∞ norm constraints on the coefficient vectors $\boldsymbol{\Xi}^{(g)}$. Additionally, physics-informed constraints could be imposed on the optimization. For example, in the case of known Sturm-Liouville form operators, constraints could be added to the optimization that directly relate $p(x)$ to its derivative $p_x(x)$. Conservation laws can also be included in the optimization to provide additional constraints, for example, on the energy within the system.

One important consideration is the practical applications of SINDy-BVP to real physical systems. Logistically, SINDy-BVP requires input data $\{\mathbf{U}, \mathbf{F}\}$, which are paired matrices of measurements of the system ($\mathbf{U}$) and different forcing functions applied to the system ($\mathbf{F}$). The matrix $\mathbf{U}$ can be constructed using a grid of sensors measuring the desired state variable ($u(x)$). In order to apply a variety of forcings, a system-specific testing jig would likely need to be constructed in which a forcing could be applied and measured simultaneously. For example, suppose the goal is to measure the thermal properties of a composite bar. A measurement jig could be constructed using an array of evenly spaced thermocouples along the bar, while the forcings could be applied to the bar using thermoelectric heaters.

The SINDy-BVP method proposed in this work successfully enables simultaneous discovery of the governing linear or nonlinear operator L of a BVP and the parametric coefficients in the operator. It extends the SINDy methodology from dynamical systems to time-invariant BVPs, and is demonstrated to work on systems with piecewise and multimodal sinusoidal parametric coefficients commonly found in heterogeneous materials systems.

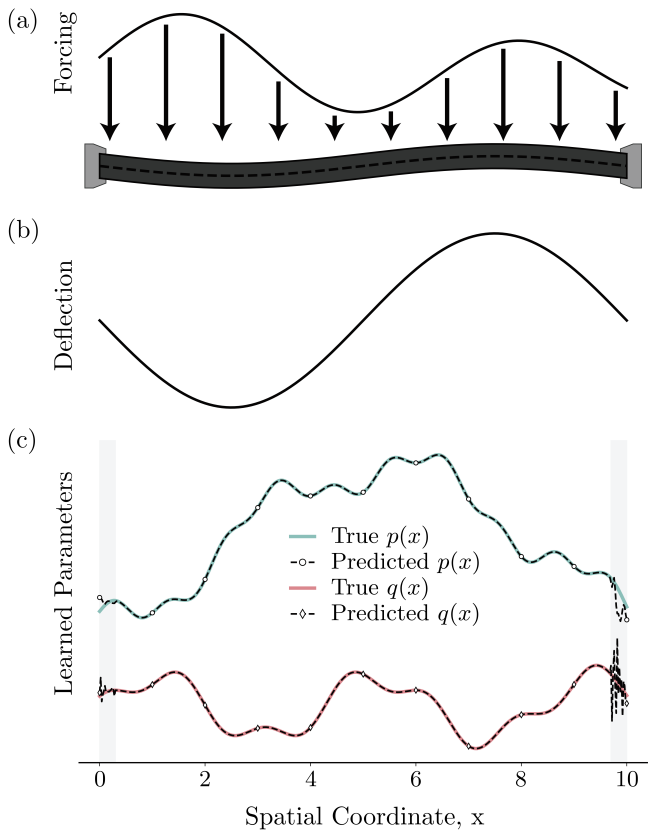


Figure 2.1: SINDy-BVP studies steady-state systems subjected to a forcing function. One simple example system is a beam clamped at both ends subjected to a static load (a). The beam deflects (b) in response to the load, and the forcing function and deflection are used for data-driven modeling via SINDy-BVP to learn the parametric coefficients (c) in the governing operator. The coefficients $p(x)$ and $q(x)$ vary spatially. The coefficients are directly related to the beam's spatially-varying mechanical properties. The grey boxes in (d) indicate that error can occur in the learned coefficients near the boundaries.

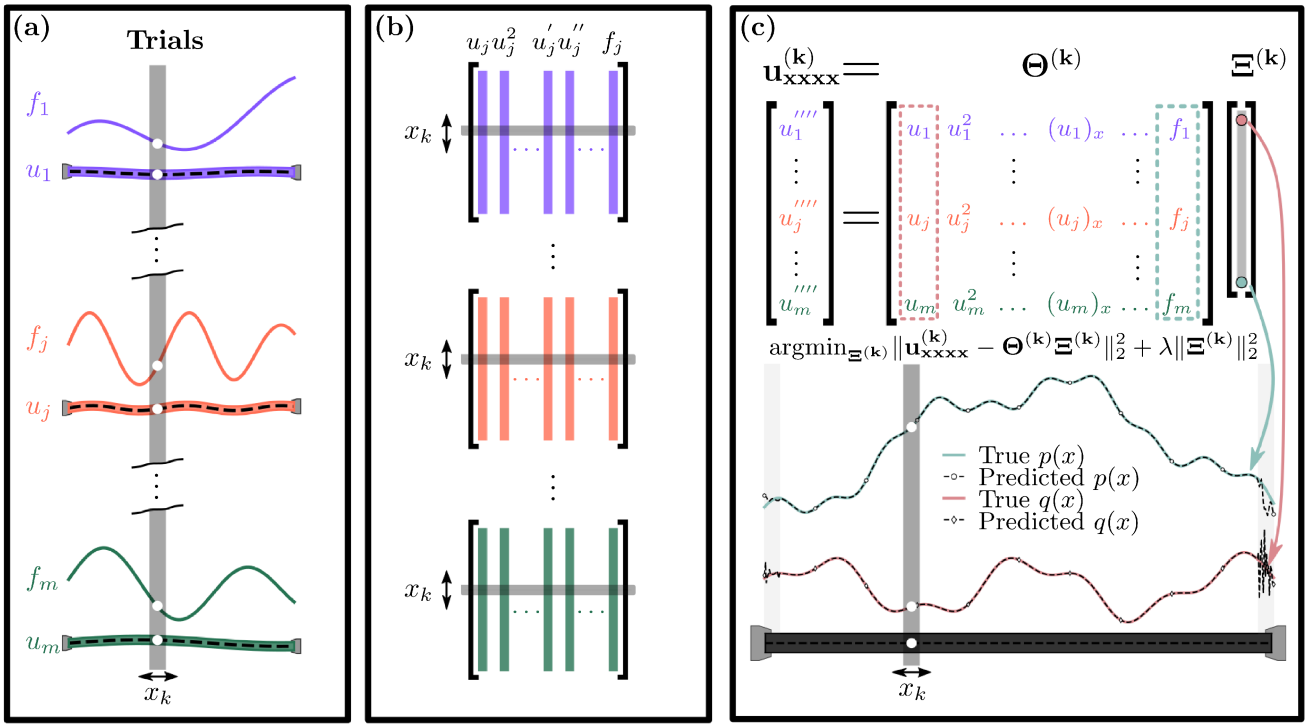


Figure 2.2: Overview of constructing the data sets and regression for each discrete spatial point in the data set. Part (a) shows a collection of trials, where different forcings ($f_j(x)$) are applied to a system yielding different responses ($u_j(x)$). A matrix containing a library of candidate terms $\mathbf{\Theta}(\mathbf{U}, \mathbf{F})$ is produced for each trial, where the rows are each a spatial point x_k and each column contains a candidate function, as seen in part (b). A sliding procedure is used to select rows of a single x_k from the libraries in part (b) to produce an aggregated library $\mathbf{\Theta}^{(k)}$ for each x_k in the data set. In (c), the regression is performed for each x_k to produce a vector of the parametric coefficients $p(x)$ and $q(x)$ at each x_k. The regression in (c) is a Ridge regression algorithm. However, a sparsity constraint is applied by an iterative thresholding and grouping mechanism of the algorithm.

Algorithm 2 Sequential Grouped Threshold Ridge Regression

Input: Candidate functions $\boldsymbol{\Theta}$, Derivatives $\mathbf{U}_{xx}$, Groups G, Regularizer λ, Threshold ϵ, Score function $r(\mathbf{x}) = \|x\|_2$, $iters$

Output: Candidate function coefficients $\boldsymbol{\Xi}$

procedure
  $\boldsymbol{\Xi} \leftarrow \text{argmin}_{\boldsymbol{\Xi}'} \|\mathbf{U}_{xx} - \boldsymbol{\Theta}\boldsymbol{\Xi}'\|_2$ ▷ Initial $\boldsymbol{\Xi}$ guess
  for $i = 1, ..., iters$ **do**
    $P \leftarrow \{g_l \in G : r(\boldsymbol{\Xi}^{(g_l)}) < \epsilon\}$ ▷ Select groups below threshold
    $\boldsymbol{\Xi}^{(P)} \leftarrow 0$ ▷ Set to zero
    $\boldsymbol{\Xi} \leftarrow \text{argmin}_{\boldsymbol{\Xi}'} \|\mathbf{U}_{xx} - \boldsymbol{\Theta}\boldsymbol{\Xi}'\|_2 + \lambda\|\boldsymbol{\Xi}\|_2$ ▷ Repeat regression
  end for
  $\boldsymbol{\Xi}^{(G)} \leftarrow \text{argmin}_{\boldsymbol{\Xi}'} \|\mathbf{U}_{xx} - \boldsymbol{\Theta}^{(G)}\boldsymbol{\Xi}'^{(G)}\|_2$ ▷ Final fit by ordinary least squares
  return $\boldsymbol{\Xi}$
end procedure

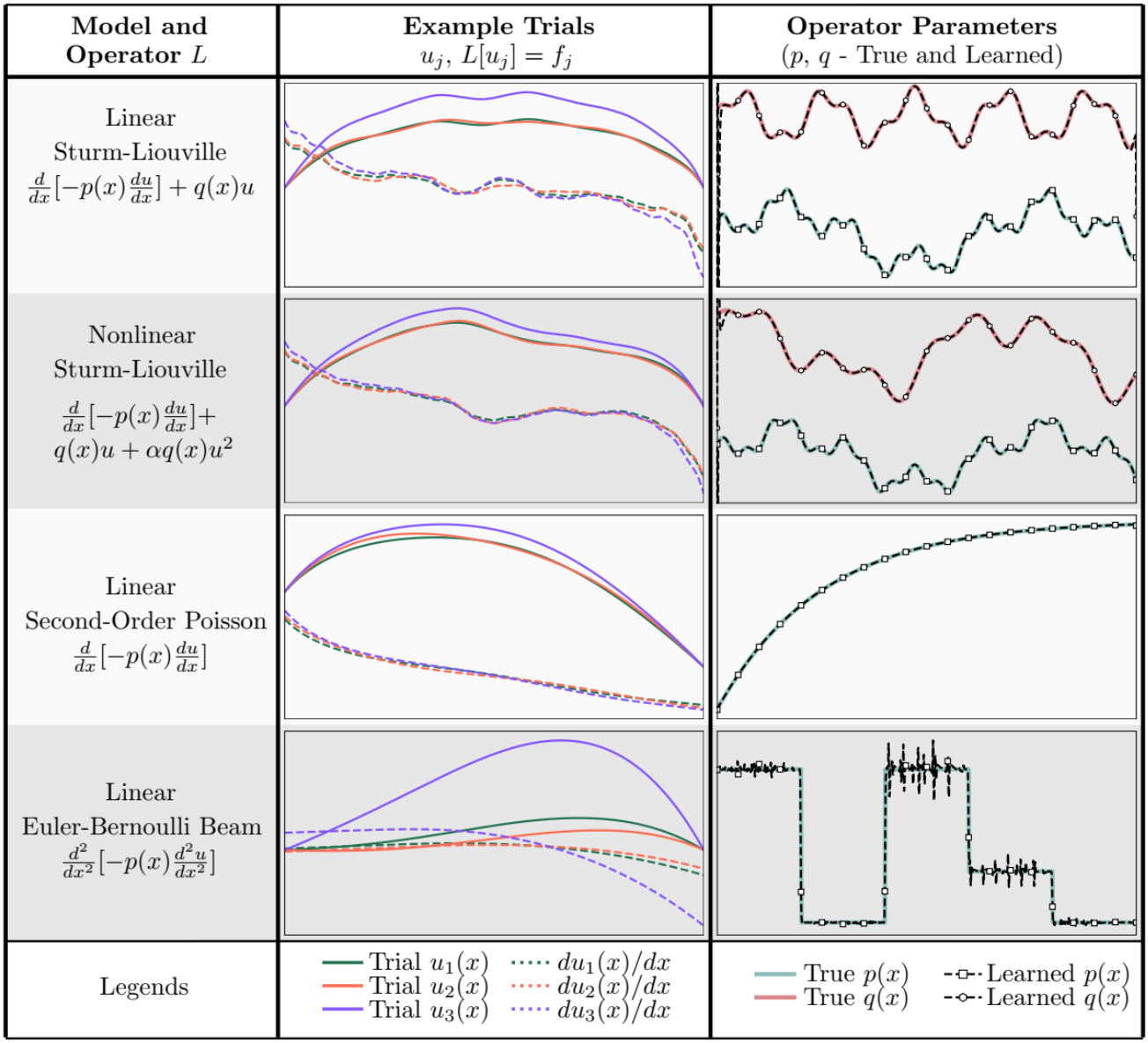


Figure 2.3: Summary of the models and operators studied with SINDy-BVP. The center column shows three example trials from the training data set ($u_1(x)$, $u_2(x)$, and $u_3(x)$). The solutions are all of $\mathcal{O}(1)$. The final column shows the parametric coefficients in the operator, as well as the inferred parameters for clean data. Coefficients $p(x)$ and $q(x)$ are plotted with an offset, with markers every 30 points.

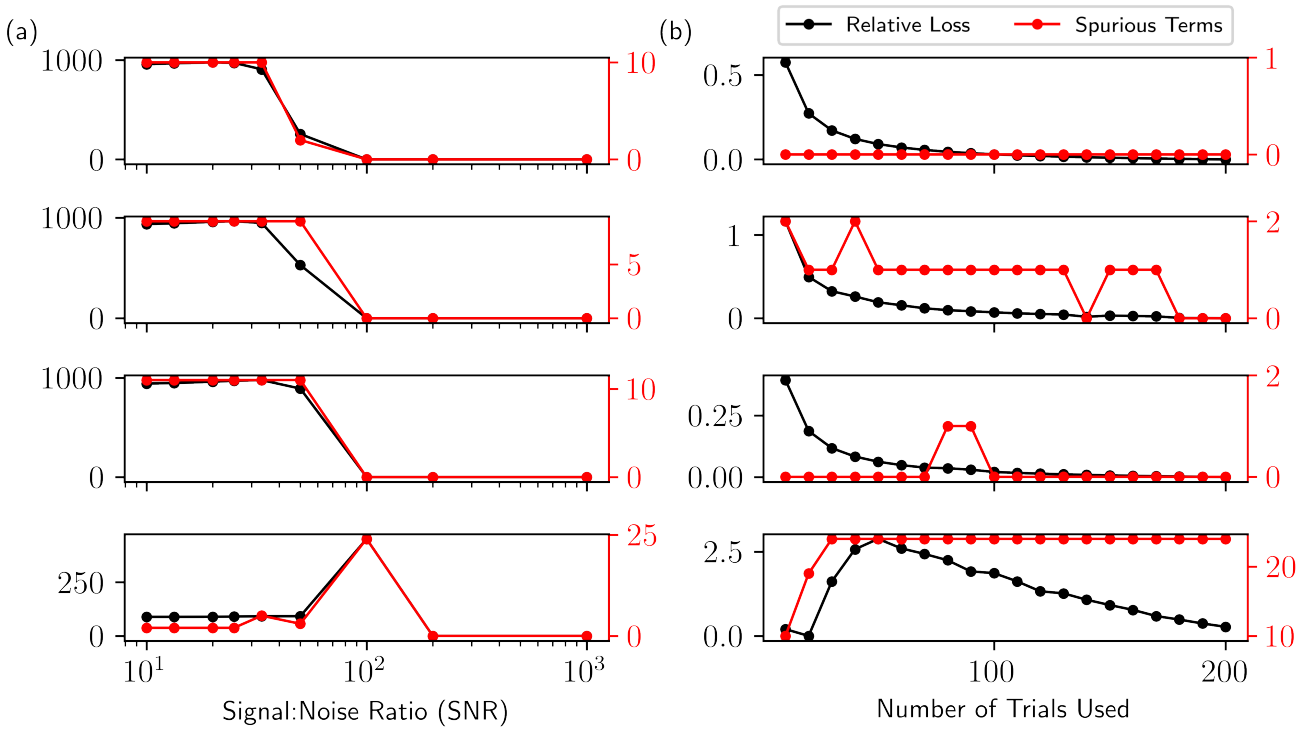


Figure 2.4: SINDy-BVP succeeds at operator identification in signals with high SNR. A collection of trials with varying SNR is used for data-driven operator identification. The plots in (a) and (b), from top to bottom, are the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, Poisson, and Euler-Bernoulli beam models. Relative loss (Equation 2.10) and a count of spurious terms in the identified model are quantified. In (a), 200 trials are used for regression and the SNR is varied. In (b), the number of trials is varied while holding a constant SNR of 100.

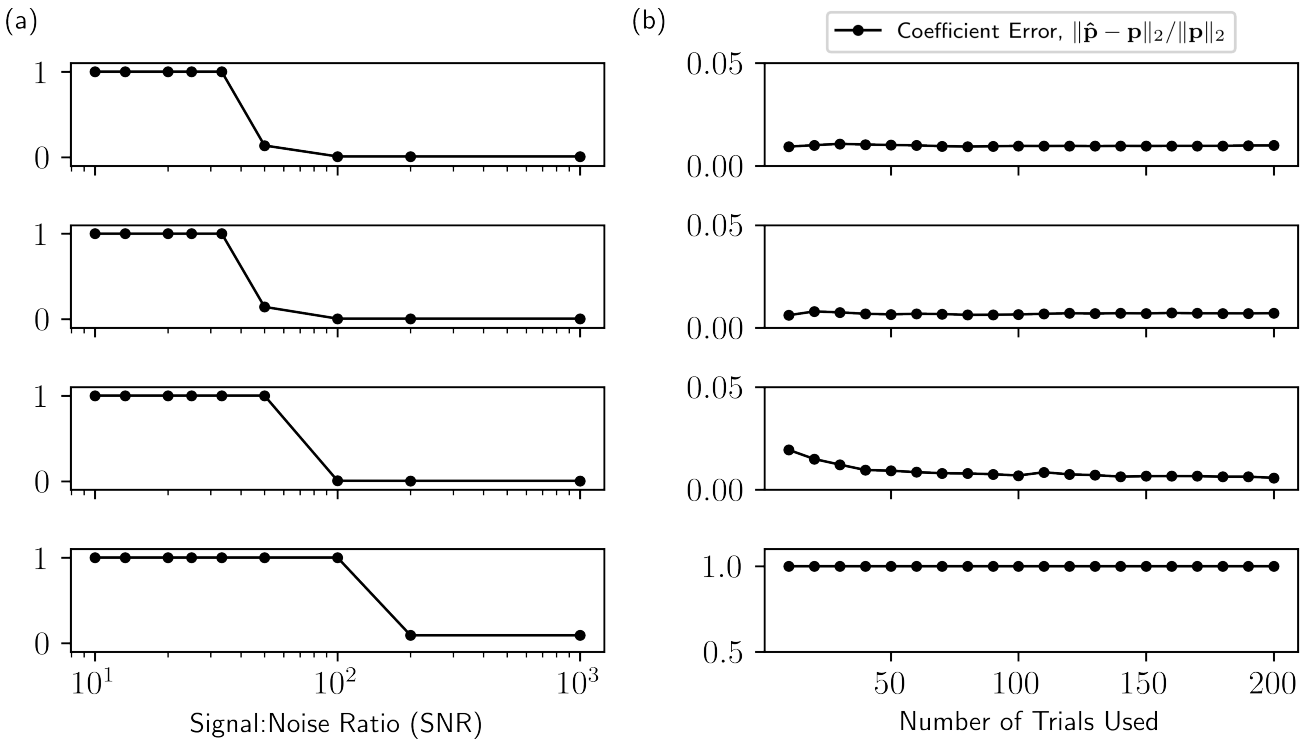


Figure 2.5: Parametric coefficient estimation in noisy data. With a known operator L, SINDy-BVP can estimate the parametric coefficients. Coefficient error is used to compare the effect of varying SNR (a) and the number of trials used as input data (b). In (a), the number of trials used as input data are held constant at 200 and in (b) the SNR is fixed at 100. The plots in (a) and (b), from top to bottom, are the Linear Sturm-Liouville, Nonlinear Sturm-Liouville, Poisson, and Euler-Bernoulli beam models.

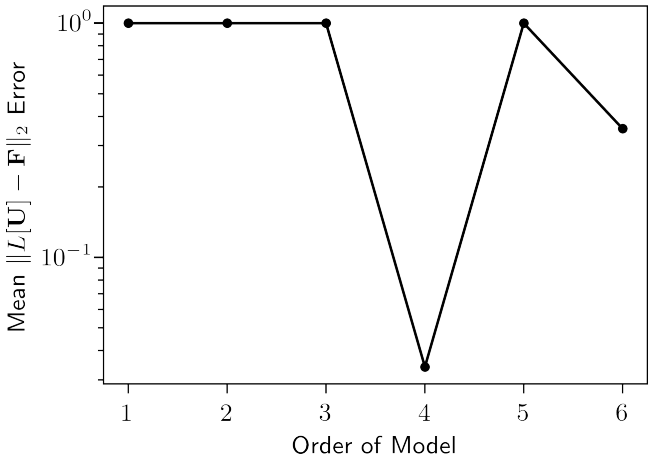


Figure 2.6: Determination of correct order equation to use for building the data-driven model. This example is the Euler-Bernoulli beam theory, which should use the left hand side term $d^4u(x)/dx^4$ to build the correct model. The model which best captures the relationship $Lu(x) = f(x)$ is determined from the validation error $\|L[\mathbf{U}_j] - \mathbf{F}_j\|_2$ from a test data set. The fourth order model ($d^4u(x)/dx^4$) exhibits the lowest error.

Chapter 3

DEEPGREEN: DEEP LEARNING OF GREEN'S FUNCTIONS FOR NONLINEAR BOUNDARY VALUE PROBLEMS

The fundamental solution, or Green's function, is a leading method for solving linear BVPs that enables facile computation of new solutions to systems under any external forcing. However, fundamental Green's function solutions for nonlinear BVPs are not feasible since linear superposition no longer holds. In this chapter, we describe a flexible deep learning approach to solve nonlinear BVPs using a dual-autoencoder architecture. The autoencoders discover an invertible coordinate transform that linearizes the nonlinear BVP and identifies both a linear operator L and Green's function G which can be used to solve new nonlinear BVPs. The method merges the strengths of the universal approximation capabilities of deep learning with the physics knowledge of Green's functions to yield a flexible tool for identifying fundamental solutions to a variety of nonlinear systems.

In Section 3.1, different approaches to similar problems are described; many of the approaches apply specifically to linearizing dynamical systems rather than time-invariant problems like the examples in this chapter. The section also describes the utility of a Green's function for solving BVPs, which is the fundamental rationale for linearizing a nonlinear BVP. Section 3.2 works through the details of the proposed technique, and describes the loss functions and network architecture. We demonstrate in Section 3.3 that the method succeeds on a variety of nonlinear systems including nonlinear Helmholtz and Sturm–Liouville problems, nonlinear elasticity, and a 2D nonlinear Poisson equation. Further discussion on the results is provided in Section

3.4.

3.1 Introduction

Boundary value problems (BVPs) are ubiquitous in the sciences [1]. From elasticity to quantum electronics, BVPs have been fundamental in the development and engineering design of numerous transformative technologies of the 20th century. Historically, the formulation of many canonical problems in physics and engineering result in *linear* BVPs: from Fourier formulating the heat equation in 1822 [7] to more modern applications such as designing chip architectures in the semi-conductor industry [40, 41]. Much of our theoretical understanding of BVPs comes from the construction of the fundamental solution of the BVP, commonly known as the Green's function [2]. The Green's function solution relies on a common property of many BVPs: *linearity*. Specifically, general solutions rely on linear superposition to hold, thus limiting their usefulness in many modern applications where BVPs are often heterogeneous and nonlinear. By leveraging modern deep learning, we are able to learn linearizing transformations of BVPs that render *nonlinear BVPs linear* so that we can construct the Green's function solution. Our deep learning of Green's functions, *DeepGreen*, provides a transformative architecture for modern solutions of nonlinear BVPs.

DeepGreen is inspired by recent works which use deep neural networks (DNNs) to discover advantageous coordinate transformations for dynamical systems [42, 43, 44, 45, 46, 47, 48, 49, 50, 51]. The universal approximation properties of DNNs [52, 53] are ideal for learning coordinate transformations that linearize nonlinear BVPs, ODEs and PDEs. Specifically, such linearizing transforms fall broadly under the umbrella of Koopman operator theory [54], which has a modern interpretation in terms of dynamical systems theory [55, 56, 57, 58]. There are only limited cases in which Koopman operators can only be constructed explicitly [59]. However *Dynamic Mode Decomposition* (DMD) [60] provides a numerical algorithm for approximating the Koopman

operator [61], with many recent extensions that improve on the DMD approximation [62]. More recently, neural networks have been used to construct Koopman embeddings [42, 44, 45, 46, 47, 48, 49, 51]. This is an alternative to enriching the observables of DMD [63, 64, 65, 66, 67, 68, 69]. Thus, neural networks have emerged as a highly effective mathematical tool for approximating complex data [70, 71] with a *linear* model. DNNs have been used in this context to discover time-stepping algorithms for complex systems [72, 73, 74, 75, 76]. Moreover, DNNs have been used to approximate constitutive models of BVPs [17].

DeepGreen leverages the success of DNNs for dynamical systems to discover coordinate transformations that linearize nonlinear BVPs so that the Green's function solution can be recovered. This allows for the discovery of the fundamental solutions for nonlinear BVPs, opening many opportunities for the engineering and physical sciences. DeepGreen exploits physics-informed learning by using autoenconders (AEs) to take data from the original high-dimensional input space to the new coordinates at the intrinsic rank of the underlying physics [42, 77, 43]. The architecture also leverages the success of *Deep Residual Networks* (DRN) [78] which enables our approach to efficiently handle near-identity coordinate transformations [51].

The Green's function constructs the solution to a BVP for any given forcing by superposition. Specifically, consider the classical linear BVP [2]

$$L[v(\mathbf{x})] = f(\mathbf{x}) \tag{3.1}$$

where L is a linear differential operator, f is a forcing, $\mathbf{x} \in \Omega$ is the spatial coordinate, and Ω is an open set. The boundary conditions $Bv(\mathbf{x}) = 0$ are imposed on $\partial\Omega$ with a linear operator B. The fundamental solution is constructed by considering the adjoint equation

$$L^{\dagger}[G(\mathbf{x}, \boldsymbol{\xi})] = \delta(\mathbf{x} - \boldsymbol{\xi}) \tag{3.2}$$

where $L^{\dagger}$ is the adjoint operator (along with its associated boundary conditions) and $\delta(\mathbf{x} - \xi)$ is the Dirac delta function. Taking the inner product of (3.1) with respect

to the Green's function gives the fundamental solution

$$v(\mathbf{x}) = (f(\boldsymbol{\xi}), G(\boldsymbol{\xi}, \mathbf{x})) = \int_{\Omega} G(\xi, \mathbf{x}) f(\boldsymbol{\xi}) d\boldsymbol{\xi}, \tag{3.3}$$

which is valid for any forcing $f(\mathbf{x})$. Thus once the Green's function is computed, the solution for arbitrary forcing functions can be easily extracted from integration. This integration represents a superposition of a continuum of delta function forcings that are used to represent $f(\mathbf{x})$.

In many modern applications, nonlinearity plays a fundamental role so that the BVP is of the form

$$N[u(\mathbf{x})] = F(\mathbf{x}) \tag{3.4}$$

where $N[\cdot]$ is a nonlinear differential operator. For this case, the principle of linear superposition no longer holds and the notion of a fundamental solution is lost. However, modern deep learning algorithms allow us the flexibility of learning a coordinate transformation (and their inverses) of the form

$$v = \boldsymbol{\psi}(u), \tag{3.5a}$$

$$f = \boldsymbol{\phi}(F), \tag{3.5b}$$

such that v and f satisfy the linear BVP (3.1) for which we generated the fundamental solution (3.3). This gives a nonlinear fundamental solution through use of this deep learning transformation.

DeepGreen is a *supervised learning* algorithm which is ultimately a high-dimensional interpolation problem [79] for learning the coordinate transformations $\boldsymbol{\psi}(u)$ and $\boldsymbol{\phi}(F)$. DeepGreen is enabled by a physics-informed deep autoencoder coordinate transformation which establishes superposition for nonlinear BVPs, thus enabling a Koopman BVP framework. The learned Green's function enables accurate construction of solutions with new forcing functions in the same way as a linear BVP. We demonstrate the DeepGreen method on a variety of nonlinear boundary value problems, including a nonlinear 2D Poisson problem, showing that such an architecture can be used

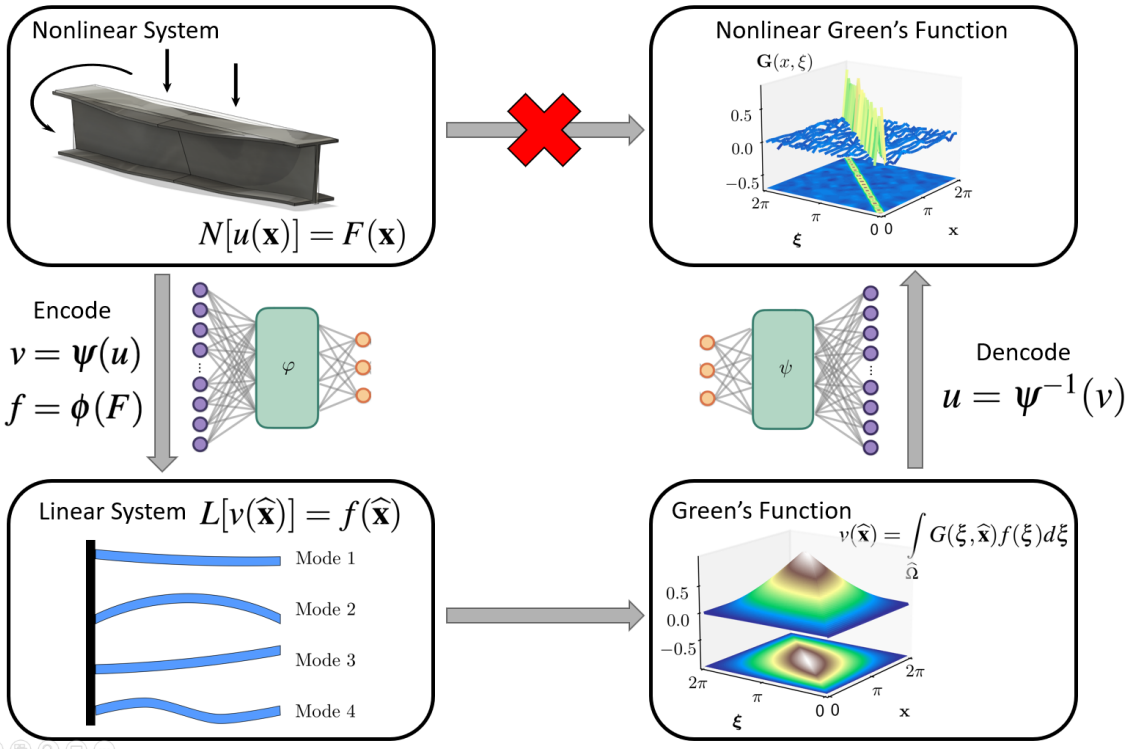


Figure 3.1: DeepGreen solves nonlinear BVPs by identifying the Green's Function of the nonlinear problem using a deep learning approach with a dual autoencoder architecture. An nonhomogenous linear BVP can be solved using the Green's function approach, but a nonlinear BVP cannot. DeepGreen transforms a nonlinear BVP to a linear BVP, solves the linearized BVP, and then inverse transforms the linear solution to solve the nonlinear BVP.

in many modern and diverse applications in aerospace, electromagnetics, elasticity, materials, and chemical reactors.

3.2 Deep Autoencoders for Linearizing BVPs

Deep AEs have been used to linearize dynamical systems, which are initial value problems. We extend this idea to BVPs. To be precise, we consider BVPs of the form

$$N[u(\mathbf{x})] = F(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{3.6a}$$

$$B[u(\mathbf{x})] = 0, \qquad \mathbf{x} \in \partial\Omega, \tag{3.6b}$$

where Ω is a simply connected open set in $\mathbb{R}^n$ with boundary $\partial\Omega$, N is a nonlinear differential operator, $F(\mathbf{x})$ is the nonhomogeneous forcing function, B is a boundary condition, and $u(\mathbf{x})$ is the solution to the BVP. We wish to find a pair of coordinate transformations of the form (3.5) such that v and f satisfy a linear BVP

$$L[v(\widehat{\mathbf{x}})] = f(\widehat{\mathbf{x}}), \qquad \widehat{\mathbf{x}} \in \widehat{\Omega}, \tag{3.7a}$$

$$\widehat{B}[v(\widehat{\mathbf{x}})] = 0, \qquad \widehat{\mathbf{x}} \in \partial\widehat{\Omega}, \tag{3.7b}$$

where L is a linear differential operator, $\widehat{\mathbf{x}}$ is the spatial coordinate in the transformed domain $\widehat{\Omega}$ with boundary $\partial\widehat{\Omega}$. Because L is linear, there is a Green's function $G(\widehat{\mathbf{x}}, \boldsymbol{\xi})$ such that the solution v to the BVP (3.7) can be obtained through convolution of the Green's function and transformed forcing function

$$v(\widehat{\mathbf{x}}) = \int_{\widehat{\Omega}} G(\xi, \widehat{\mathbf{x}}) f(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{3.8}$$

The coordinate transformation along with the Green's function of the linearized BVP provide the analog of a Green's function for the nonlinear BVP (3.6). In particular, for a forcing function $F(\mathbf{x})$, the transformed forcing function is $f = \boldsymbol{\phi}(F)$. The solution to the linearized BVP can be obtained using the Green's function $v = \int G(\boldsymbol{\xi}, \widehat{\mathbf{x}}) f(\boldsymbol{\xi}) d\boldsymbol{\xi}$. Then the solution to the nonlinear BVP (3.6) is obtained by inverting the coordinate transformation $u = \boldsymbol{\psi}^{-1}(v)$ to obtain the solution to the nonlinear BVP, $u(\mathbf{x})$.

The question that remains is how to discover the appropriate coordinate transformations $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$. We leverage the universal approximation properties of neural networks in order to learn these transformations. In order to use neural networks, we first need to discretize the BVP. Let $\mathbf{u}$ be a spatial discretization of $u(\mathbf{x})$ and $\mathbf{F}$ be a

discretization of $F(\mathbf{x})$. Then the discretized version of the BVP (3.6) is

$$\mathbf{N}[\mathbf{u}] = \mathbf{F}, \tag{3.9a}$$

$$\mathbf{B}[\mathbf{u}] = \mathbf{0}. \tag{3.9b}$$

Neural networks $\boldsymbol{\psi}_u$ and $\boldsymbol{\phi}_F$ are used to transform $\mathbf{u}$ and $\mathbf{F}$ to the latent space vectors $\mathbf{v}$ and $\mathbf{f}$

$$\mathbf{v} = \boldsymbol{\psi}_u(\mathbf{u}), \tag{3.10a}$$

$$\mathbf{f} = \boldsymbol{\phi}_F(\mathbf{F}), \tag{3.10b}$$

where $\mathbf{v}$ and $\mathbf{f}$ satisfy the linear equation

$$\mathbf{L}\mathbf{v} = \mathbf{f}, \tag{3.11}$$

for some matrix $\mathbf{L}$, which is also learned. In order to learn invertible transforms $\boldsymbol{\psi}_u$ and $\boldsymbol{\phi}_F$, we construct the problem as a pair of autoencoder networks.

In this construction, the transforms $\boldsymbol{\psi}_u$ and $\boldsymbol{\phi}_F$ are the encoders and the transform inverses are the decoders. The network architecture and loss functions are shown in Figure 3.2. The neural network is trained using numerous and diverse solutions to the nonlinear BVP (3.9), which can be obtained with many different forcings $\mathbf{F}_k$. Consider a dataset comprised of pairs of discretized solutions and forcing functions $\{\mathbf{u}_k, \mathbf{F}_k\}_{k=1}^N$. The loss function for training the network is the sum of six losses, each of which enforces a desired condition. The loss functions can be split into three categories:

1. **Autoencoder losses:** We wish to learn invertible coordinate transformations given by equations (3.10a) and (3.10b). In order to do so, we use two autoencoders. The autoencoder for $\mathbf{u}$ consists of an encoder $\boldsymbol{\psi}_u$ which performs the transformation (3.10a) and a decoder $\boldsymbol{\psi}_u^{-1}$ which inverts the transformation. In

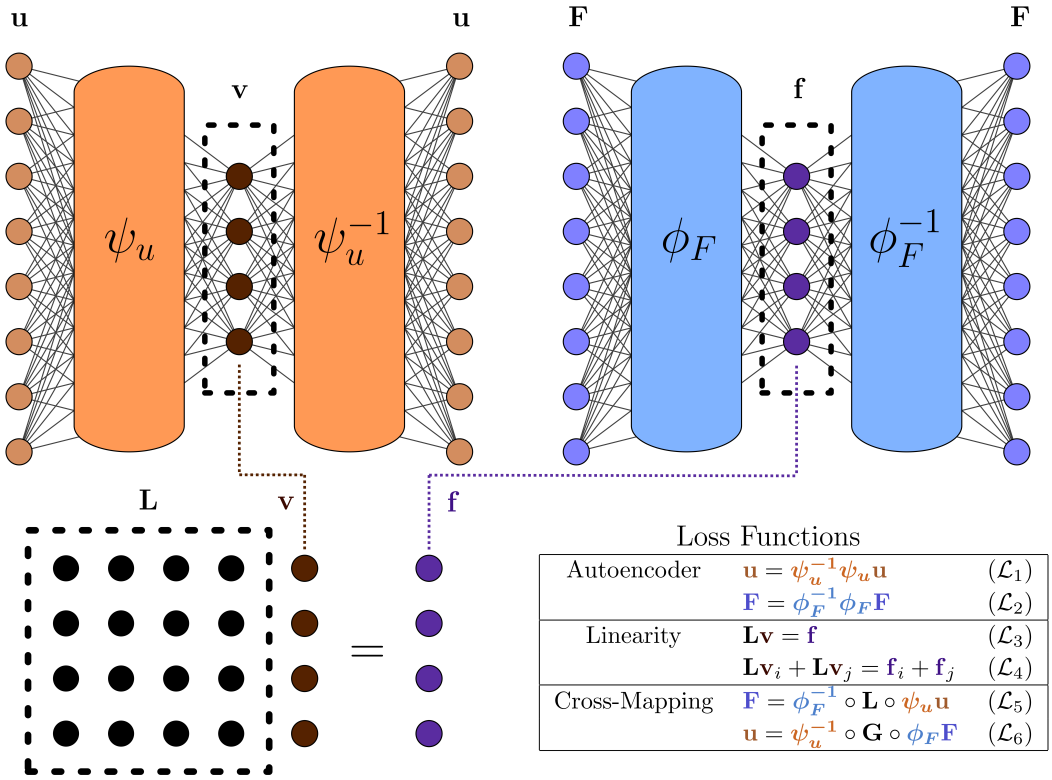


Figure 3.2: DeepGreen architecture. Two autoencoders learn invertible coordinate transformations that linearize a nonlinear boundary value problem. The latent space is constrained to exhibit properties of a linear system, including linear superposition, which enables discovery of a Green's function for nonlinear boundary value problems.

order to enforce that the encoder and decoder are inverses, we use the autoencoder loss

$$\mathcal{L}_1 = \frac{1}{N}\sum_{k=1}^{N} \frac{\left\|\mathbf{u}_k - \boldsymbol{\psi}_u^{-1} \circ \boldsymbol{\psi}_u(\mathbf{u}_k)\right\|_2^2}{\|\mathbf{u}_k\|_2^2}. \tag{3.12}$$

Similarly, there is an autoencoder for $\mathbf{F}$ where the encoder $\boldsymbol{\phi}_F$ performs the transformation (3.10b). This transformation also has an inverse enforced by the associated autoencoder loss function

$$\mathcal{L}_2 = \frac{1}{N}\sum_{k=1}^{N} \frac{\left\|\mathbf{F}_k - \boldsymbol{\phi}_F^{-1} \circ \boldsymbol{\phi}_F(\mathbf{F}_k)\right\|_2^2}{\|\mathbf{F}_k\|_2^2}. \tag{3.13}$$

2. **Linearity losses:** In the transformed coordinate system, we wish for the BVP

to be linear so that the operator can be represented by a matrix $\mathbf{L}$. The matrix $\mathbf{L}$ and the encoded vectors $\mathbf{v}$ and $\mathbf{f}$ should satisfy equation (3.11). This is enforced with the linear operator loss

$$\mathcal{L}_3 = \frac{1}{N}\sum_{k=1}^{N}\frac{\|\mathbf{f}_k - \mathbf{L}\mathbf{v}_k\|_2^2}{\|\mathbf{f}_k\|_2^2}. \tag{3.14}$$

The major advantage of working with a linear operator is that linear superposition holds. We use a linear superposition loss in order to further enforce the linearity of the operator in the latent space

$$\mathcal{L}_4 = \frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N}\frac{\|(\mathbf{f}_i + \mathbf{f}_j) - \mathbf{L}(\mathbf{v}_i + \mathbf{v}_j)\|_2^2}{\|\mathbf{f}_i + \mathbf{f}_j\|_2^2}. \tag{3.15}$$

3. **Cross-mapping losses:** The losses described above are theoretically sufficient to find coordinate transformations for $\mathbf{u}$ and $\mathbf{F}$ as well as a linear operator $\mathbf{L}$. However, in practice the two autoencoders were not capable of generating the Green's function solution. To rectify this, we add two "cross-mapping" loss functions that incorporate parts of both autoencoders. The first cross-mapping loss enforces the following mapping from $\mathbf{u}$ to $\mathbf{F}$. First, one of the solutions from the dataset $\mathbf{u}_k$ is encoded with $\boldsymbol{\psi}_u$. This is an approximation for $\mathbf{v}_k$. This is then multiplied by the matrix $\mathbf{L}$, giving an approximation of $\mathbf{f}_k$. Then the result is decoded with $\boldsymbol{\phi}_F^{-1}$. This gives an approximation of $\mathbf{F}_k$. The $\mathbf{u}$ to $\mathbf{F}$ cross-mapping loss is given by the formula

$$\mathcal{L}_5 = \frac{1}{N}\sum_{k=1}^{N}\frac{\left\|\mathbf{F}_k - \boldsymbol{\phi}_F^{-1}\circ\mathbf{L}\circ\boldsymbol{\psi}_u(\mathbf{u}_k)\right\|_2^2}{\|\mathbf{F}_k\|_2^2}. \tag{3.16}$$

We can similarly define a cross-mapping from $\mathbf{F}$ to $\mathbf{u}$. For a forcing function $\mathbf{F}_k$ from the dataset, it is encoded with $\boldsymbol{\phi}_F$, multiplied by the Green's function ($\mathbf{G} = \mathbf{L}^{-1}$), and then decoded with $\boldsymbol{\psi}_u^{-1}$ to give an approximation of $\mathbf{u}_k$. The $\mathbf{F}$ to $\mathbf{u}$ cross-mapping loss is

$$\mathcal{L}_6 = \frac{1}{N}\sum_{k=1}^{N}\frac{\left\|\mathbf{u}_k - \boldsymbol{\psi}_u^{-1}\circ\mathbf{L}^{-1}\circ\boldsymbol{\phi}_F(\mathbf{F}_k)\right\|_2^2}{\|\mathbf{u}_k\|_2^2}. \tag{3.17}$$

Note that this final loss function gives the best indication of the performance of the network to solve the nonlinear BVP (3.9) using the Green's function. The strategy for solving (3.9) for a given discrete forcing function $\mathbf{F}$ is to encode the forcing function to obtain $\mathbf{f} = \boldsymbol{\phi}_F(\mathbf{F})$, apply the Green's function as in equation (3.8) to obtain $\mathbf{v}$, and then decode this function to get the solution $\mathbf{u} = \boldsymbol{\psi}_u^{-1}(\mathbf{v})$. The discrete version of the convolution with the Green's function given in equation (3.8) is multiplication by the matrix $\mathbf{L}^{-1}$.

For the encoders $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ and decoders $\boldsymbol{\phi}^{-1}$ and $\boldsymbol{\psi}^{-1}$, we use a residual neural network (ResNet) architecture [78]. The ResNet architecture has been successful in learning coordinate transformations for physical systems [51] and is motivated by near-identity transformations in physics. The linear operator $\mathbf{L}$ is constrained to be a real symmetric matrix and therefore is self-adjoint. Additionally, $\mathbf{L}$ is initialized as the identity matrix. Therefore, $\mathbf{L}$ is strictly diagonally dominant for at least the early parts of training which guarantees $\mathbf{L}$ is invertible and well-conditioned.

3.3 Results

The DeepGreen architecture, which is highlighted in Fig. 3.2 and whose detailed loss functions are discussed in the last section, is demonstrated on a number of canonical nonlinear BVPs. The first three BVPs are one-dimensional systems and the final one is a two-dimensional system. The nonlinearities in these problems do not allow for a fundamental solution, thus recourse is typically made to numerical computations to achieve a solution. DeepGreen, however, can produce a fundamental solution which can then be used for any new forcing of the BVP.

3.3.1 Cubic Helmholtz

The architecture and methodology is best illustrated using a basic example problem. The example problem uses a nonhomogeneous second-order nonlinear Sturm–Liouville model with constant coefficients and a cubic nonlinearity, thus making it a cubic

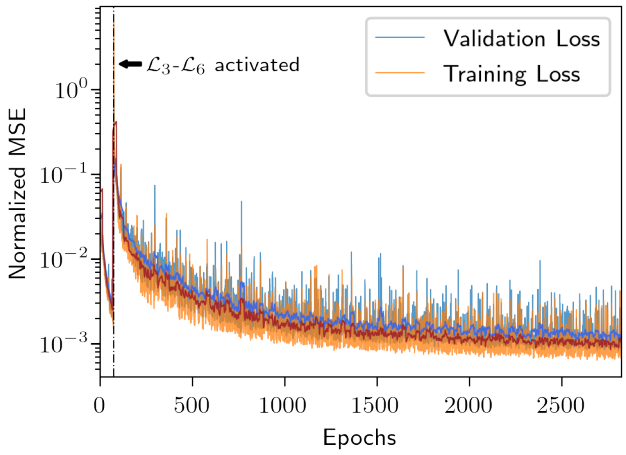


Figure 3.3: Learning curve. This is a typical learning curve for the DeepGreen architecture. The vertical dashed line indicates where the training procedure transitions from autoencoders-only (only $\mathcal{L}_1$ and $\mathcal{L}_2$) to a full-network training procedure (all losses).

Helmholtz equation. The differential equation is given by

$$u'' + \alpha u + \epsilon u^3 = F(x), \tag{3.18a}$$

$$u(0) = u(2\pi) = 0, \tag{3.18b}$$

where $u = u(x)$ is the solution when the system is forced with $F(x)$ with $x \in (0, 2\pi)$, $\alpha = -1$ and $\epsilon = -0.3$. The notation u'' denotes $\frac{d^2}{dx^2}u(x)$. The dataset contains discretized solutions and forcings, $\{\mathbf{u}_k, \mathbf{F}_k\}_{k=1}^N$. The data is divided into three groups: training, validation, and test. The training and validation sets are used for training the model. The test set is used to evaluate the results. The training set contains $N_{train} = 8906$ vector pairs $\mathbf{u}_k$ and $\mathbf{F}_k$. The validation set contains $N_{validation} = 2227$ and test set contains $N_{test} = 1238$.

Training the Model

The autoencoders used in this example are constructed with fully connected layers. In both autoencoders, a ResNet-like identity skip connection connects the input layer

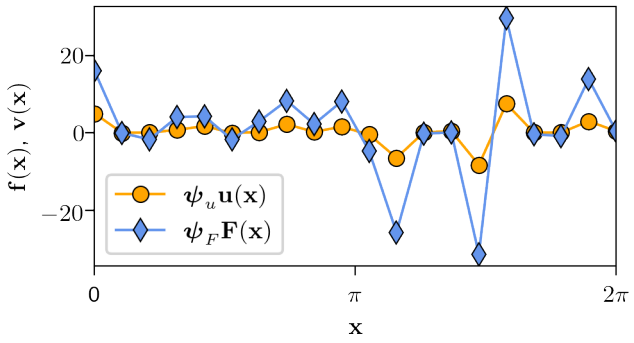


Figure 3.4: Latent space representations $\mathbf{v}_k$ and $\mathbf{f}_k$. The autoencoder transformation $\boldsymbol{\psi}_u$ encodes $\mathbf{u}_k$ to the latent space, producing the vector $\mathbf{v}_k$ (orange). The forcing vector $\mathbf{F}_k$ is transformed by $\boldsymbol{\psi}_F$ to the encoded vector $\mathbf{f}_v$ (blue).

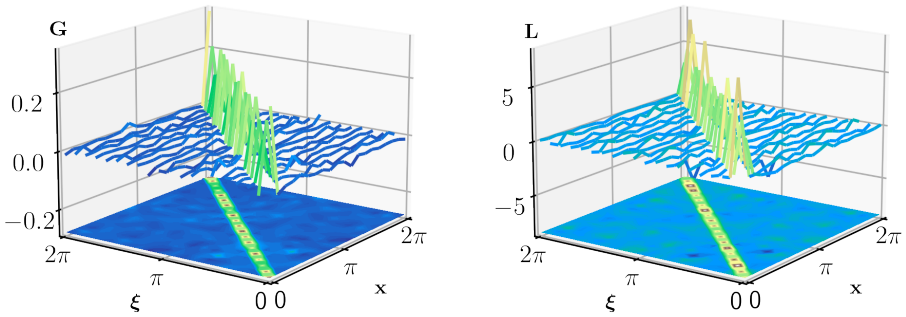


Figure 3.5: Visualized operator and Green's function. Discovered Green's function $\mathbf{G} = \mathbf{L}^{-1}$ and corresponding linear operator $\mathbf{L}$.

to the layer before dimension reduction in the encoder, and the first full-dimension layer in the decoder with the final output layer (see Figure **??**).

The model is trained in a two-step procedure. First, the autoencoders are trained, without connection in the latent space, to condition the networks as autoencoders. In this first phase, only the autoencoder loss functions listed in Figure 3.2 are active ($\mathcal{L}_1$ and $\mathcal{L}_2$). After a set number of epochs, the latent spaces are connected by an invertible matrix operator, $\mathbf{L}$, and the remaining 4 loss functions in Figure 3.2 become active ($\mathcal{L}_3$–$\mathcal{L}_6$). In the final phase of training, the autoencoder learns to encode a latent

space representation of the system where properties associated with linear systems hold true, such as linear superposition.

Figure 3.3 shows a typical training loss curve. The vertical dashed line indicates the transition between the two training phases. The models in this work are trained for 75 epochs in the first autoencoder-only phase and 2750 epochs in the final phase. The first-phase epoch count was tuned empirically based on final model performance. The final phase epoch count was selected for practical reasons; the training curve tended to flatten around 2750 epochs in all of our tested systems. The autoencoder latent spaces are critically important. The latent space is the transformed vector space where linear properties (e.g. superposition) are enforced which enables the solution of nonlinear problems. In the one-dimensional problems, the latent spaces vectors $\mathbf{v}$ and $\mathbf{f}$ are in $\mathbb{R}^{20}$.

The latent spaces did not have any obvious physical interpretation, and qualitatively appeared similar to the representations shown in Figure 3.4. We trained 100 models to check the consistency in the learned model and latent space representations, but discovered the latent spaces varied considerably. This implies the existence of an infinity of solutions to the coordinate transform problem, which indicates further constraints could be placed on the model.

Despite lacking obvious physical interpretations, the latent space enables discovery of an invertible operator $\mathbf{L}$ which described the linear system $\mathbf{L}[\mathbf{v}_k] = \mathbf{f}_k$. The operator matrix $\mathbf{L}$ can be inverted to yield the Green's function matrix $\mathbf{G}$, which allows computation of solutions to the linearized system $\mathbf{v}_k = \mathbf{G}[\mathbf{f}_k]$. An example of the operator $\mathbf{L}$ and its inverse $\mathbf{G}$ are shown in Figure 3.5. The operator and Green's function shown in Figure 3.5 display an important prominent feature seen in all of the results: a diagonally-dominant structure. We initialize the operator as an identity matrix, but the initialization had little impact on the diagonally-dominant form of the learned operator and Green's function matrices. The diagonally-dominant operators indicate that the deep learning network tends to discover a coordinate transform yielding a

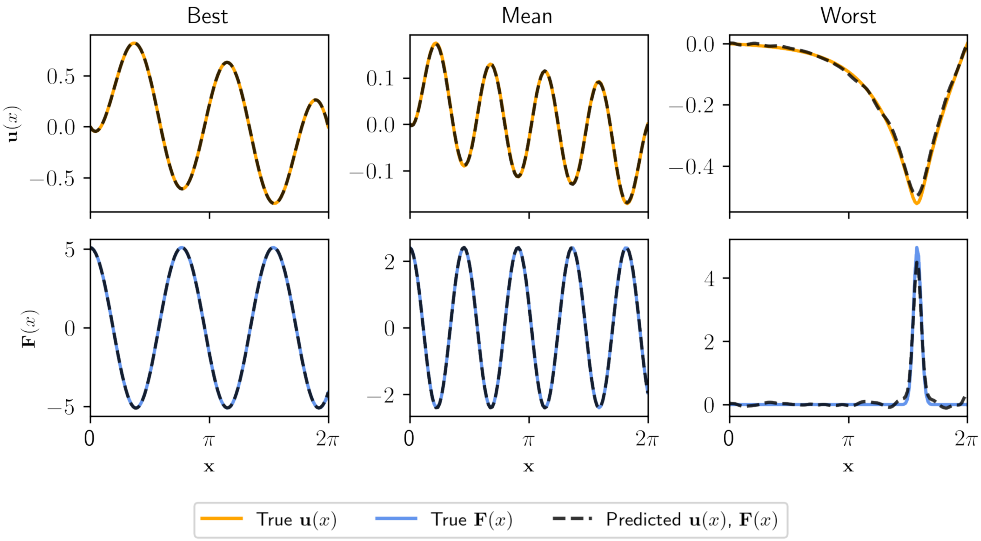


Figure 3.6: Model predictions on test data. The top row shows the true solution $\mathbf{u}_k(x)$ and the solution predicted by the network given the forcing $\mathbf{F}_k(x)$ using the Green's function $\mathbf{G}$. The bottom row shows the true forcing function $\mathbf{F}_k(x)$ compared to the forcing computed by applying the operator $\mathbf{L}$ to the solution $\mathbf{u}_k$. Three columns show the best, mean, and worst case samples as evaluated by the sum of normalized $\ell 2$ reconstruction errors.

nearly-orthonormal basis, which mirrors the common approach of diagonalization in spectral theory for Hermitian operators. Furthermore, diagonally-dominant matrices guarantee favorable properties for this application such as being well-conditioned and non-singular.

We emphasize that training parameters and model construction choices used in this work were not extensively optimized. We expect the model performance can be improved in a myriad of ways including extending training times, optimizing model architecture, modifying the size of the latent spaces, restricting the form of the operator, and applying additional constraints to the model. However, these topics are not the main scope of the present work; our focus is to illustrate the use of autoencoders as a coordinate transform for finding solutions to nonlinear BVPs.

Evaluating the Model

The goal for this model is to find a Green's function $\mathbf{G}$ for computing solutions $\mathbf{u}_k$ to a nonlinear BVP governed by (3.6) for a given forcing function $\mathbf{F}_k$. Similarly, we can estimate the forcing term, $\mathbf{F}_k$, given the solution $\mathbf{u}_k$. The model is consequently evaluated by its ability to use the learned Green's function and operator for predicting solutions and forcings, respectively, for new problems from a withheld test data set.

Recall the original model is trained on data where the forcing function is a cosine or Gaussian function. As shown in Figure 3.6, the model performs well on withheld test data where the forcing functions are cosine or Gaussian functions, producing a cumulative loss around 10^{-4}. The solutions $\mathbf{u}_k$ and forcing $\mathbf{F}_k$ are depicted for the best, mean, and worst samples scored by cumulative loss.

It's important to note the test data used in Figure 3.6 is similar to the training and validation data. Because ML models typically work extremely well in interpolation problems, it is reasonable to expect the model to perform well on this test data set.

As an interesting test to demonstrate the ability of the model to extrapolate, we prepared a separate set of test data $\{\mathbf{u}_k, \mathbf{F}_k\}_{k=1}^{N}$ containing solutions where $\mathbf{F}_k$ are cubic polynomial forcing functions. This type of data was not present in training, and provides some insight into the generality of the learned linear operator and Green's function matrices. Figure 3.7 shows examples of how the model performs on these cubic polynomial-type forcing functions. Similar to Figure 3.6, the best, mean, and worst samples are shown as graded by overall loss. Figures 3.6 and 3.7 provide some qualitative insight into the model's performance on specific instances selected from the pool of evaluated data. A quantitative perspective of the model's performance is presented in Figure 3.8. This box plot shows statistics (median value, Q_1, Q_3, and range) for four of the loss functions evaluated on the similar (cosine and Gaussian) test data. Note the superposition loss function is *not* scored in this plot because the superposition loss function can only be evaluated within a single batch, and the loss

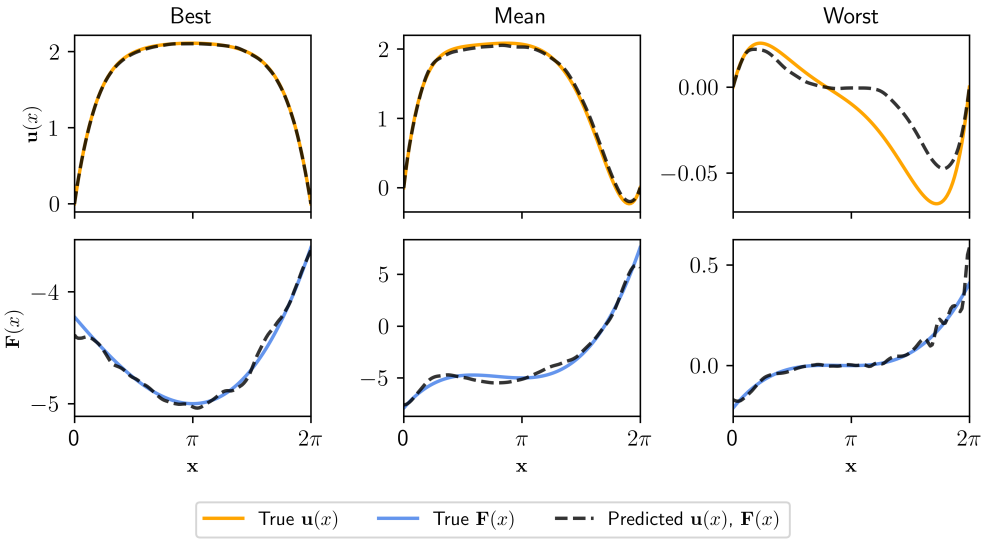


Figure 3.7: Model predictions on cubic Helmholtz forced system. The top row shows the true solution $\mathbf{u}_k(x)$ and the solution predicted by the network given the forcing $\mathbf{F}_k(x)$ using the Green's function $\mathbf{G}$. The bottom row shows the true forcing function $\mathbf{F}_k(x)$ compared to the forcing computed by applying the operator $\mathbf{L}$ to the solution $\mathbf{u}_k$. Three columns show the best, mean, and worst case samples as evaluated by the sum of normalized $\ell 2$ reconstruction errors.

depends on batch size and composition.

In conclusion, the DeepGreen architecture enables discovery of invertible, linearizing transformations that facilitate identification of a linear operator and Green's function to solve nonlinear BVPs. It is tested on data similar and dissimilar to the training data, and evaluated on the loss functions that guide the training procedure. The discovered operator and Green's function take on a surprisingly diagonally-dominant structure, which hints at the model's preference to learn an optimal basis. The model appears to extrapolate beyond the test data, suggesting that the learned operator is somewhat general to the system.

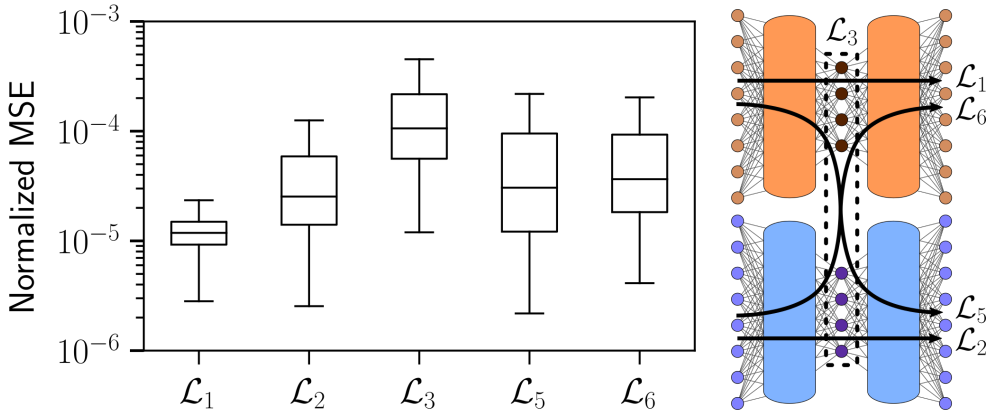


Figure 3.8: Model performance summary. Distribution of loss values are shown for every sample in the test data set. Model loss functions are minimized during training, making them a natural metric to use for summarizing performance.

3.3.2 Nonlinear Sturm–Liouville and Biharmonic Operators

In addition to the example system described above, the approach was applied to two other one-dimensional systems. We used the same training procedure and forcing functions that were described in Section 3.3.1. The first is a system governed by the nonlinear Sturm–Liouville equation

$$[-p(x)u']' + q(x)(u + \epsilon u^3) = F(x),$$
$$u(0) = u(2\pi) = 0,$$

where $\epsilon = 0.4$ controls the extent of nonlinearity, and $p(x)$ and $q(x)$ are spatially-varying coefficients

$$p(x) = 0.5\sin(x) - 3,$$
$$q(x) = 0.6\sin(x) - 2,$$

with $x \in [0, 2\pi]$. The final one-dimensional system is a biharmonic operator with an added cubic nonlinearity

$$[-pu'']'' + q(u + \epsilon u^3) = F(x),$$
$$u(0) = u(2\pi) = u'(0) = u'(2\pi) = 0,$$

where $p = -4$ and $q = 2$ are the coefficients and $\epsilon = 0.4$ controls the nonlinearity. As in the prior example, the forcing functions in the training data are cosine and Gaussian functions.

Results for all the one-dimensional models, including the cubic Helmholtz example from Section 3.3.1, are presented in Table 3.1. Model performance is quantitatively summarized by box plots and the Green's function matrix is shown for each model.

Importantly, the learned Green's function matrices consistently exhibit diagonally-dominant structure. The losses for the nonlinear cubic Helmholtz equation and the nonlinear Sturm–Liouville equation are similar which indicates that spatially-varying coefficients do not make the problem significantly more difficult for the DeepGreen architecture. In contrast, the loss for the nonlinear biharmonic equation are about an order of magnitude higher than the other two systems. This result implies the fourth-order problem is more difficult than the second-order problems. Also of note is that the linear operator loss $\mathcal{L}_3$ is consistently the highest loss across all models. Therefore, it is easier for DeepGreen to find invertible transformations for the solutions and forcing functions than it is to find a linear operator that connects the two latent spaces.

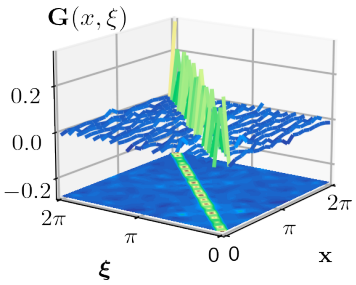
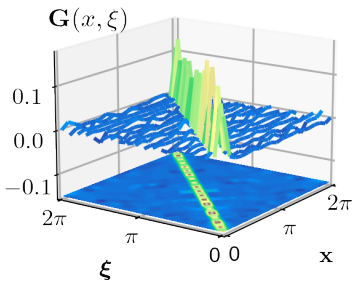
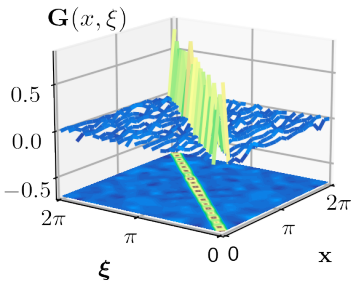
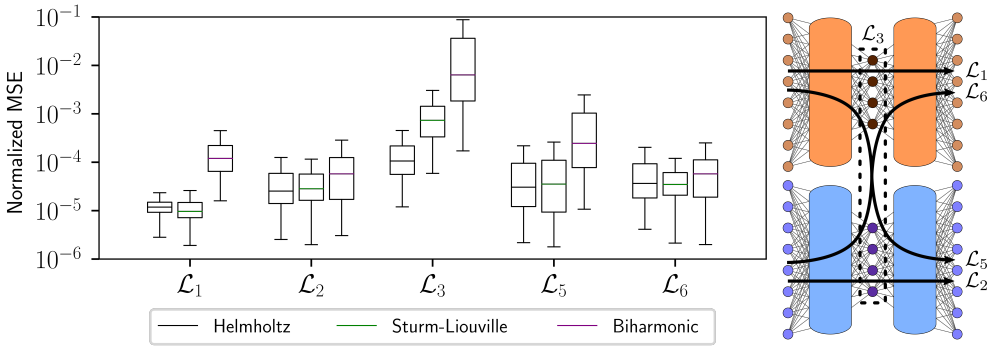
Nonlinear cubic Helmholtz (constant coefficients) $u'' + \alpha u + \epsilon u^3 = F$ $u(0) = u(L) = 0$	Nonlinear Sturm–Liouville (varying $p(x)$, $q(x)$) $[-pu']' + qu + \alpha q u^3 = F$ $u(0) = u(L) = 0$	Nonlinear Biharmonic operator (constant coefficients) $-pu'''' + qu + \alpha q u^3 = F$ $u(0) = u(L) = 0$
		
		

Table 3.1: Summary of results for three one-dimensional models. The models are provided with the Green's function learned by DeepGreen. A summary box plot shows the relative losses $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_5$, and $\mathcal{L}_6$ for all three model systems.

3.3.3 Nonlinear Poisson Equation

We also tested our method on a two-dimensional system. The two-dimensional model is a nonlinear version of the Poisson equation with Dirichlet boundary conditions

$$-\nabla \cdot \left[(1+u^2)\nabla u\right] = F(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{3.19a}$$

$$u = 0, \qquad \mathbf{x} \in \partial\Omega, \tag{3.19b}$$

where $\Omega := (0, 2\pi) \times (0, 2\pi)$. Similar to the one-dimensional models, the forcing functions used to train the model are cosine and Gaussian functions. The sizes of the data sets are also similar to the one-dimensional data sets. The training data contains $N_{train} = 9806$ vector pairs $\mathbf{u}_k$ and $\mathbf{F}_k$, the validation data contains $N_{validation} = 2452$, and the test data contains $N_{test} = 1363$.

The network architecture of the encoders and decoders for the two-dimensional example differs from the one-dimensional examples. Instead of fully connected layers, convolutional layers were used in the encoders and decoders. However, we still use a ResNet architecture. Additionally, the latent space vectors are in $\mathbb{R}^{200}$. Note that the method proposed for discovering Green's functions allows for any network architecture to be used for the encoders and decoders. For the one-dimensional example, similar results were obtained using fully connected and convolutional layers. However, the convolutional architecture was better in the two-layer case and also allowed for a more manageable number of parameters for the wider network that resulted from discretizing the two-dimensional space.

The operator and Green's function for the two-dimensional model are similar to those displayed in shown in Figure 3.5. The diagonal dominance is even more prevalent in this case than the one-dimensional example. The model was evaluated on test data containing cosine and Gaussian forcing functions. Figure 3.9a shows the true solution $\mathbf{u}(x)$ and forcing function $\mathbf{F}(x)$ as well as the network predictions for the example from the test data for which the model performed the best (i.e. the smallest value of the loss). The difference between the true and predicted functions is shown in the right column of Figure 3.9a and is scaled by the infinity norm of the true solution or forcing functions. Figure 3.9b shows similar results but for the worst example from the test data. In both cases, the model gives a qualitatively correct solution for both $\mathbf{u}(x)$ and $\mathbf{F}(x)$. Unsurprisingly, the network struggles most on highly localized forcing functions and has the highest error in the region where the forcing occurs.

The model was also evaluated on test data that has cubic polynomial forcing

functions, a type of forcing function not found in the training data. The best and worst examples are shown in Figure 3.10. Although the model does not perform as well for test data which is not similar to the training data, the qualitative features of the predicted solutions are still consistent with the true solutions. Figure 3.11 shows a box plot of the model's performance on the similar (cosine and Gaussian forcing test data). The results are similar to the one-dimensional results, and, in fact, better than the biharmonic operator model.

3.4 Conclusion

We have leveraged the expressive capabilities of deep learning to discover linearizing coordinates for nonlinear BVPs, thus allowing for the construction of the *fundamental solution or nonlinear Green's function*. Much like the Koopman operator for time-dependent problems, the linearizing transformation provides a framework whereby the fundamental solution of the linear operator can be constructed and used for any arbitrary forcing. This provides a broadly applicable mathematical architecture for constructing solutions for nonlinear BVPs, which typically rely on numerical methods to achieve solutions. Our DeepGreen architecture can achieve solutions for arbitrary forcings by simply computing the convolution of the forcing with the Green's function in the linearized coordinates.

Given the critical role that BVPs play in the mathematical analysis of constrained physical systems subjected to external forces, the DeepGreen architecture can be broadly applied in nearly every engineering discipline since BVPs are prevalent in diverse problem domains including fluid mechanics, electromagnetics, quantum mechanics, and elasticity. Importantly, DeepGreen provides a bridge between a classic and widely used solution technique to nonlinear BVP problems which generically do not have principled techniques for achieving solutions aside from brute-force computation. DeepGreen establishes this bridge by providing a transformation which allows linear superposition to hold. DeepGreen is a flexible, data-driven, deep learning approach to

solving nonlinear boundary value problems (BVPs) using a dual-autoencoder architecture. The autoencoders discover an invertible coordinate transform that linearizes the nonlinear BVP and identifies both a linear operator L and Green's function G which can be used to solve new nonlinear BVPs. We demonstrated that the method succeeds on a variety of nonlinear systems including nonlinear Helmholtz and Sturm–Liouville problems, nonlinear elasticity, and a 2D nonlinear Poisson equation. The method merges the strengths of the universal approximation capabilities of deep learning with the physics knowledge of Green's functions to yield a flexible tool for identifying fundamental solutions to a variety of nonlinear systems.

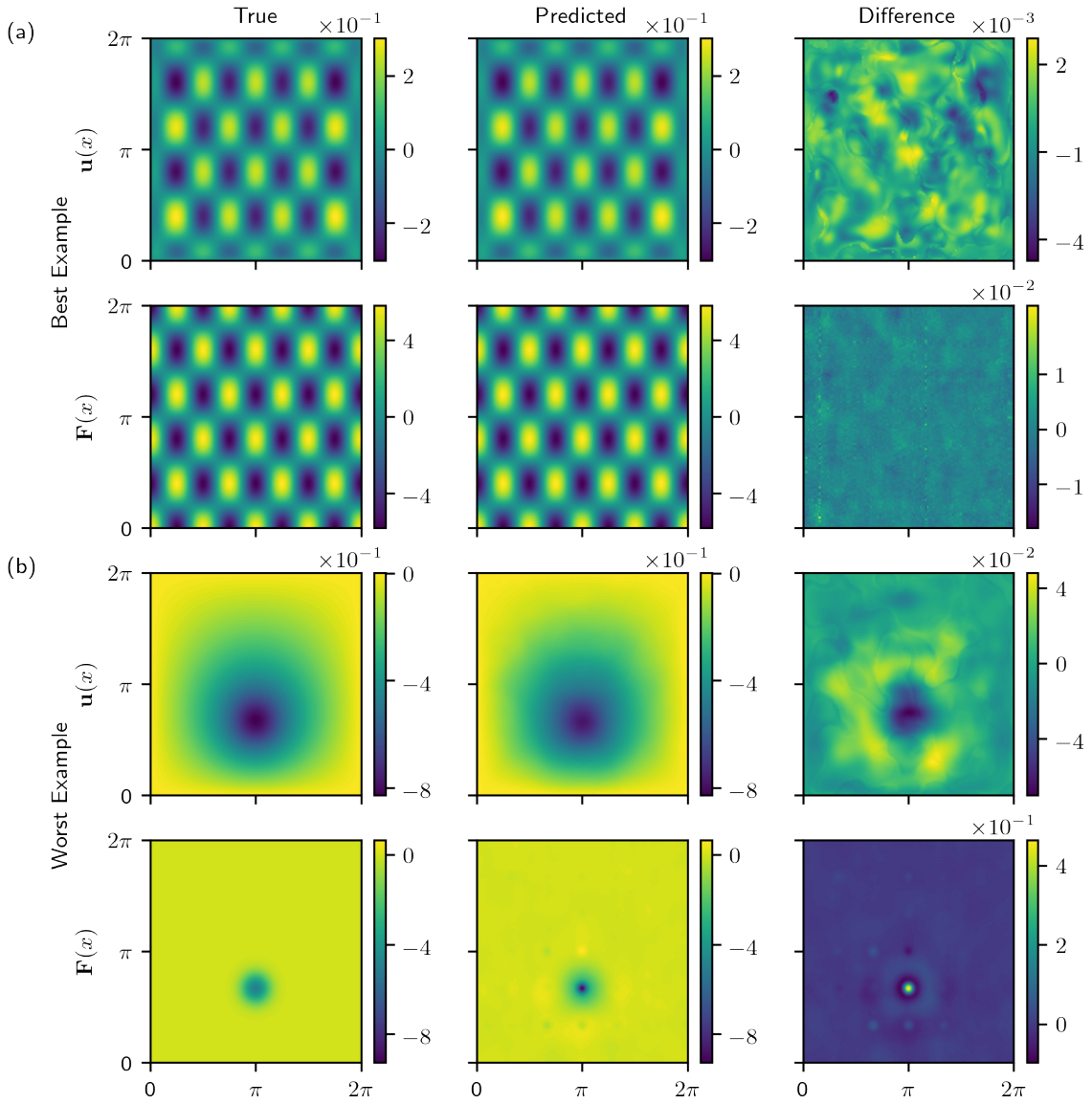


Figure 3.9: Model predictions for the (a) best and (b) worst examples from test data with Gaussian and cosine forcings. In both (a) and (b), the top row shows the true solution $\mathbf{u}(\mathbf{x})$, the predicted solution using the Green's function, and the difference between the true and predicted solution. The bottom row shows the true forcing function $\mathbf{F}(\mathbf{x})$, the predicted forcing function, and the difference between the true and predicted forces. In order to account for the difference in scale between $\mathbf{u}(\mathbf{x})$ and $\mathbf{F}(\mathbf{x})$, the differences are scaled by the infinity norm of the true solution or forcing function (Difference $=$ (True $-$ Predicted)$/||\text{True}||_\infty$).

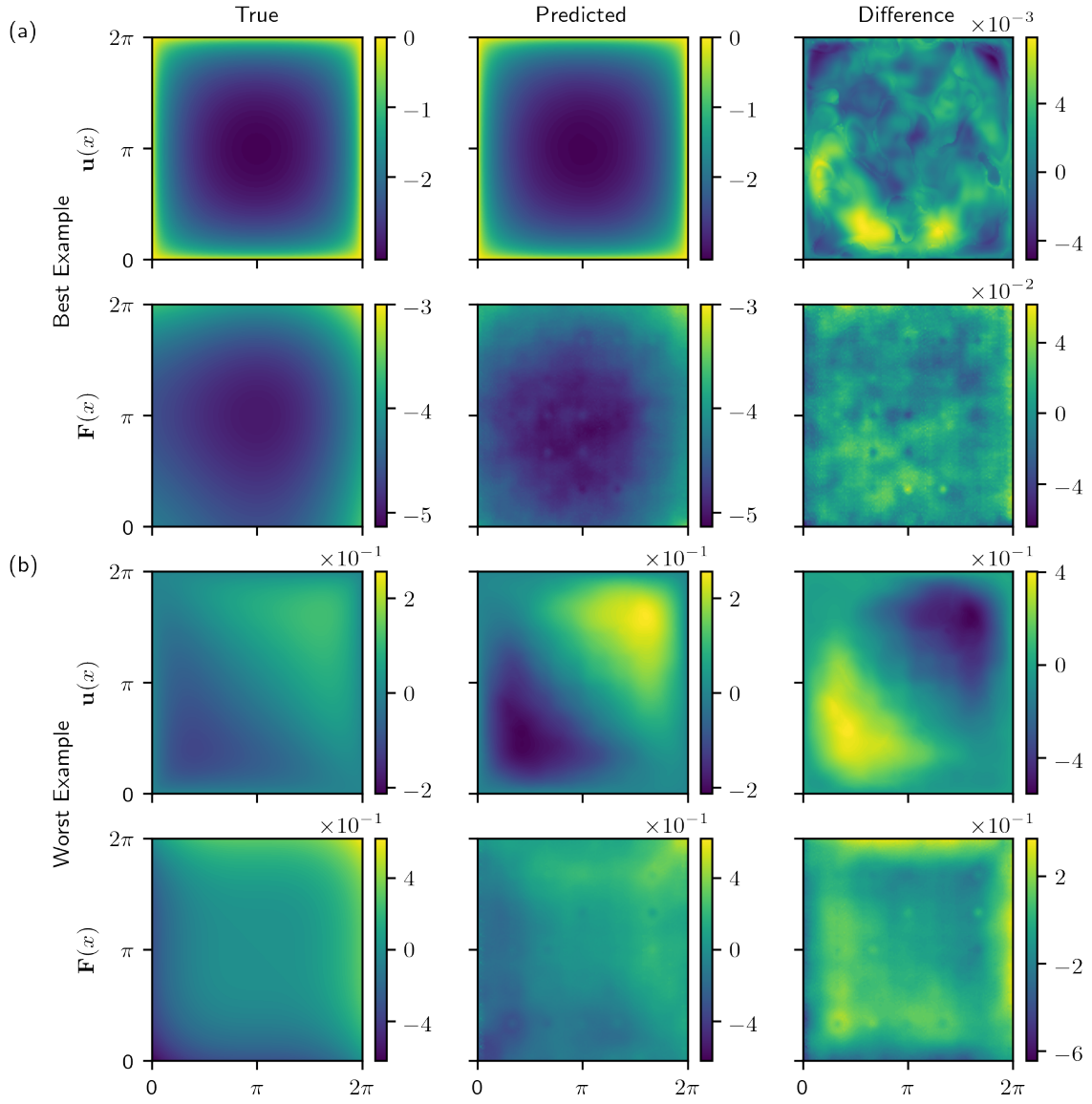


Figure 3.10: Model predictions for the (a) best and (b) worst examples from test data with cubic polynomial forcings. In both (a) and (b), the top row shows the true solution $\mathbf{u}(\mathbf{x})$, the predicted solution using the Green's function, and the difference between the true and predicted solution. The bottom row shows the true forcing function $\mathbf{F}(\mathbf{x})$, the predicted forcing function, and the difference between the true and predicted forces. In order to account for the difference in scale between $\mathbf{u}(\mathbf{x})$ and $\mathbf{F}(\mathbf{x})$, the differences are scaled by the infinity norm of the true solution or forcing function (Difference = (True − Predicted)/$||\text{True}||_\infty$).

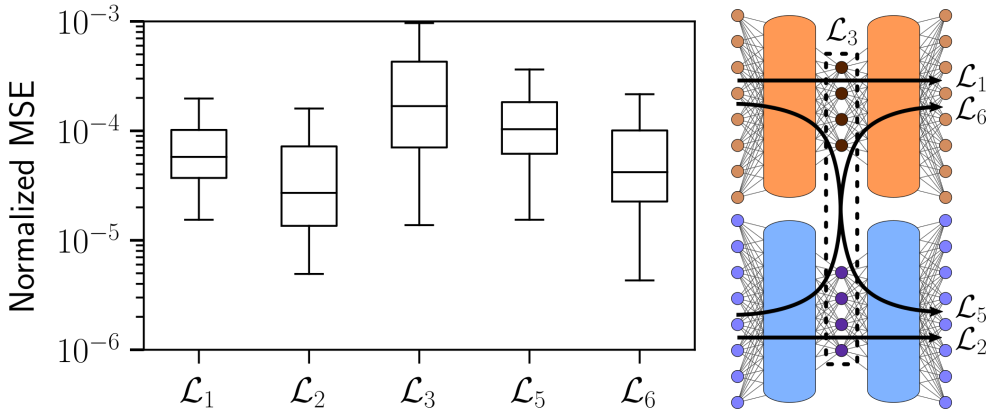


Figure 3.11: Two-dimensional Poisson model performance summary. Distribution of loss values are shown for every sample in the test data set. Model loss functions are minimized during training, making them a natural metric to use for summarizing performance.

Chapter 4

EXTRACTION OF INSTANTANEOUS FREQUENCIES AND AMPLITUDES IN NONSTATIONARY TIME-SERIES DATA

Time-series analysis is critical for a diversity of applications in science and engineering. By leveraging the strengths of modern gradient descent algorithms, the Fourier transform, multi-resolution analysis, and Bayesian spectral analysis, we propose a data-driven approach to time-frequency analysis that circumvents many of the shortcomings of classic approaches, including the extraction of instantaneous parameters in nonstationary signals with discontinuities in their behavior. The method introduced is equivalent to a *nonstationary Fourier mode decomposition* (NFMD) for nonstationary and nonlinear temporal signals, allowing for the accurate identification of instantaneous frequencies and amplitudes of oscillatory modes in the data.

In Section 4.1, historical and modern related works in time-frequency analysis are discussed. Section 4.2 provides details on the algorithm, including the computation of instantaneous parameters from the outputs of the decomposition. The method is demonstrated on a diversity of time-series data, including synthetic test systems and experimental data from cantilever-based electrostatic force microscopy, in Section 4.3. Concluding remarks are provided in Section 4.4.

4.1 Introduction

Time series data analysis is ubiquitous and foundational in scientific analysis and engineering model design [3]. Indeed, it has revolutionized nearly every scientific discipline by enabling the development of test models for observed natural phenomena

in diverse applications that include planetary motion, chemical reactions, meteorological patterns, and transport phenomena. In a typical scientific workflow, observations are made on a system and fit to a time series model, which can include classical methods from statistics, such as ARIMA (autoregressive integrated moving average) and its variants [3], or more recent neural network based approaches [71, 80], such as LSTM [81] (long-term, short-term memory), GRU (gated recurrent units) [82], and echo-state networks [83]. These diverse mathematical strategies regress to models fit to historical training data, often making assumptions that the data is generated from a stationary process with Gaussian distributed statistics. However, this workflow is often complicated by observations with non-Guassian noise, the existence of nonstationary processes, and/or nonlinear system dynamics. These challenges make forecasting exceptionally difficult, requiring the re-training of models as new data becomes available. By integrating elements of modern gradient descent algorithms, the Fourier transform, multi-resolution analysis, and Bayesian spectral analysis [84], we can train an interpretable Fourier mode-based model for analyzing nonstationary signals with periodic components, thus circumventing the challenges normally associated with nonstationary processes and allowing for accurate identification of instantaneous frequencies and their amplitudes.

Joseph Fourier revolutionized time series analysis with the introduction of his eponymous transform in 1822, which he developed while studying heat conduction [7]. The transform empowered an understanding of the frequency-energy spectrum of time-series signals and spurred the development of Fourier transform-powered time-frequency analyses [85]. These spectrum-based analysis tools have been extensively applied in systems exhibiting periodic and quasi-periodic behaviors, including oscillators and waves. So extensive are the applications that the field of *harmonic analysis* has emerged as a consequence. Harmonic analysis has been applied to a diverse range of problems spanning many size and time scales, including mechanical vibrations and machine health monitoring [86], speech and music recognition [87, 88], oceanographic

tide modeling [89], telecommunications and power systems [90], and quantum mechanics [91].

Despite its widespread use and generality, the Fourier transform has a number of restrictions that limit its utility in analyzing nonlinear and nonstationary processes [92]. This was recognized by Denis Gabor in considering radar technologies of the mid-20th century [93]. Indeed, Gabor suggested circumventing these issues in part by using short-time, or windowed, Fourier transforms. This led to improvements in time-frequency analysis and eventually to the development of wavelet theory [94]. Gabor and wavelet transforms provide rich visualizations of time-frequency representations with *spectrograms* and *scalograms* respectively [36]. More recently, the Fourier transform has inspired a new class of time-frequency methods, detailed below, that complement traditional Fourier transform-based approaches[95, 96, 97, 98, 99, 100]. Among them, the Hilbert-Huang transform (HHT) [92] has become a common tool for understanding nonstationary processes [101, 102, 103]. The HHT combines empirical mode decomposition (EMD), which separates a multi-component signal into simpler periodic modes called intrinsic mode functions (IMFs), with Hilbert spectral analysis. Hilbert spectral analysis leverages the Hilbert transform to compute the analytic signal of each periodic mode identified by EMD. In turn, the analytic signal can be leveraged to compute the instantaneous phase, instantaneous frequency, and instantaneous amplitude of the input signal. A significant theoretical framework has been developed around applying the Hilbert transform to vibrational problems [104, 105, 106]. However, the algorithm behind the EMD is empirical, thus lacking a broader theoretical underpinning.

Our method is complementary with a newer generation of time-frequency analysis algorithms that address specific shortcomings of classic approaches. The Tycoon method was introduced to handle signals with extremely fast changing frequencies [98], but focuses on signals that are (relatively) stationary. The variational mode decomposition (VMD) aims to simultaneously identify periodic modes with time-

dependent non-linear phase functions by maximizing the smoothness of the amplitude of the periodic modes [95]. VMD has been successfully applied to a broad array of problems, though the importance placed on smooth amplitude functions limits its applications for situations with discontinuities in modes' phase or amplitude functions. A wide array of approaches have been built on the VMD and extend it to different types of signals and multichannel measurements [107, 96, 86]. A different Fourier mode-based algorithm has also been developed which uses an approach similar to basis pursuit [97]. Other approaches focus on decomposition of specific models, such as harmonic oscillators, and the parameters that describe them. These approaches include both Hilbert vibrational decomposition and Kalman filter-based approaches [108, 109, 110, 102] along with sparsity-promoting decompositions [99, 100]. Wigner distribution-based approaches show great promise for decomposing multicomponent signals with modes that have crossover between frequencies [111].

In this work, we develop a method for extracting the instantaneous frequencies and amplitudes from time-series data. The methods is equivalent to a *nonstationary Fourier mode decomposition* (NFMD) for nonstationary and nonlinear temporal signals. Importantly, it produces interpretable signal decompositions that can handle signals with multiple periodic components, non-linear phase functions, and sharp discontinuities in the phase function or periodic mode amplitudes. Adopting the work of Lange et al [75], which employed a similar architecture for future state prediction rather than interpretable time-frequency analysis, the proposed method leverages modern gradient descent optimization to fit temporally-local linear Fourier modes. The approach resembles the *short time Fourier transform* (STFT) wherein smaller temporal segments of the signal are analyzed independently, and the resulting analyses are combined to provide a full time-frequency representation of the signal. The NFMD fits Fourier modes to each signal segment, and computes the mode frequency and amplitude for each signal segment through a gradient descent optimization with a nonlinear Fourier basis objective function. The method results in a superior time-

frequency analysis to the HHT for nonstationary signals, and improves both temporal and spatial resolutions compared to the STFT. The NFMD can be applied to systems with fast-changing frequencies and abrupt changes in the signal mean, such as machine health monitoring [112], seismology [113], vibration-based imaging modalities [114, 101], neurochemical and biochemical signals [115], and photonic sensing [116, 117].

4.2 Methods

The NFMD analysis proposed combines elements of modern gradient descent algorithms, the Fourier transform, Bayesian spectral analysis, and an algorithm similar to STFT to learn an interpretable Fourier mode-based model for analyzing nonstationary signals with periodic components. In general, we aim to fit a model $\mathbf{y}_t$ to a measured signal $\mathbf{z}_t$. We begin by framing the Fourier mode decomposition approach for an entire time series signal. The subsequent section shows how the method is combined with a segment-by-segment analysis, reminiscent of STFT, to propose a full time-frequency analysis. Finally, the instantaneous signal parameters and nonstationary part of the signal are addressed. We present a simple algorithm for computing the nonstationary signal component from the learned Fourier mode representations.

4.2.1 Fourier Mode Decomposition

Consider the time series data $\mathbf{z}_t = \{\mathbf{z}(t_1), \mathbf{z}(t_2), \cdots, \mathbf{z}(t_n)\}$ sampled at n discrete evenly-spaced times in $t \in \mathbb{R}^+$. The signal is assumed to be periodic or quasi-periodic. The most common frequency-domain signal analysis method is the Fourier transform, which allows the input signal $\mathbf{z}_t$ to be represented as a Fourier series. The series representation is a sum of sines and cosines that provides insight into the frequency-energy spectrum of the signal.

The Fourier transform, typically implemented as the Fast Fourier Transform (FFT), assumes a periodic input signal that satisfies the relationship $\mathbf{z}(t_i) = \mathbf{z}(t_i + P)$, where

Algorithm 3 Fourier mode signal decomposition (FMD)

Input: Signal $\mathbf{z}_t$, Initial frequency guess $\boldsymbol{\omega}$, Error tolerance *tol*

Output: Learned frequency $\boldsymbol{\omega}$, Learned amplitude $\mathbf{A}$, Residual error $E(\mathbf{A}, \boldsymbol{\omega})$

procedure
  $\mathbf{A} \leftarrow \mathbf{z}_t(\Omega(\boldsymbol{\omega}t))^{-1}$ ▷ Amplitude from initial frequency guess
  while $E(\mathbf{A}, \boldsymbol{\omega}) > tol$ **do** ▷ Use gradient descent to optimize frequency vector $\boldsymbol{\omega}$
    $\boldsymbol{\omega} \leftarrow \text{argmin}_{\boldsymbol{\omega}^*} E(A, \boldsymbol{\omega}^*)$
    $\mathbf{A} \leftarrow \mathbf{z}_t(\Omega(\boldsymbol{\omega}t))^{-1}$
    Update $E(\mathbf{A}, \boldsymbol{\omega})$
  end while
  return $\boldsymbol{\omega}$, A, $E(A, \boldsymbol{\omega})$
end procedure

P is the period of the periodic data. This periodicity assumption imparts a limited frequency resolution, F_s/P, that scales linearly with the sampling frequency of the signal, F_s, and inversely with the period P. Note the period P is typically set to the total time $(t_n - t_1)$ of the discrete input signal.

To overcome these limitations of the FFT, we adapt the Fourier series representation and exploit some of its mathematical properties to enable the NFMD time series analysis. First, the Fourier series model is framed in a more general context which uses a finite number of modes and flexible mode frequencies which need to be determined. The model

$$\mathbf{y}_t(t) = \mathbf{y}_t = \sum_{k=1}^{K} F_k(t) = \sum_{k=1}^{K} a_k \cos(\omega_k t) + b_k \sin(\omega_k t)$$

is used, where $F_k(t)$ is the general Fourier mode function, the coefficients a_k and b_k are the weights for the cosine and sine components of each mode, respectively, ω_k is

the frequency of mode k, and K modes are considered. In matrix form, the model is

$$\mathbf{y}_t(t) = \mathbf{A}\Omega(\boldsymbol{\omega}t) = \begin{bmatrix} a_1 & \dots & a_K & b_1 & \dots & b_K \end{bmatrix} \begin{bmatrix} \cos(\omega_1 t) \\ \vdots \\ \cos(\omega_K t) \\ \sin(\omega_1 t) \\ \vdots \\ \sin(\omega_K t) \end{bmatrix}, \tag{4.1}$$

where $\mathbf{A} \in \mathbb{R}^{2\times K}$ is a vector of the coefficients, and $\Omega(\boldsymbol{\omega}t) \in \mathbb{R}^{2\times K}$ is a vector of cosines and sines with frequency $\boldsymbol{\omega} \in \mathbb{R}^K$. The vector $\mathbf{A}$ is determined as the optimal coefficient vector given a frequency vector $\boldsymbol{\omega}^*$, by fitting to the time-series data $\mathbf{z}_t$, using the computation

$$\mathbf{A} = \mathbf{z}_t(\Omega(\boldsymbol{\omega}^* t))^{-1}.$$

This is a common approach for discovering component amplitudes in Bayesian spectral analysis [84]. The vector $\boldsymbol{\omega} = [\omega_1, \dots, \omega_K]$ is determined by the optimization

$$\text{minimize} \quad E(\mathbf{A}, \boldsymbol{\omega}) = \sum_{t\in[t_1,t_n]} (\mathbf{z}_t - \mathbf{A}\Omega(\boldsymbol{\omega}t))^2. \tag{4.2}$$

This method has been previously demonstrated, including the theory behind the optimization [75]. This approach does *not* appear to be a convex optimization objective, given the nonlinear objective with cosine and sine functions in $\Omega(\cdot)$, and therefore should not yield globally optimal solutions. Although this objective does lack global convexity, this pitfall is avoided by using initial guesses near the optimal solutions by leveraging the FFT for an initial guess. The initial guess frames this problem on an error surface that is locally convex, and therefore allows for highly accurate frequency estimation by gradient descent. Ultimately, this allows our input signal to be fit to a model that is a superposition of Fourier modes with superior frequency resolution to traditional Fourier transforms. The *Fourier mode decomposition* (FMD) algorithm for decomposing a full time series signal $\mathbf{z}_t$ is presented in Algorithm 3. This algorithm enables the time-frequency analysis framework described in the next section.

4.2.2 Nonstationary Fourier Mode Decomposition

The NFMD algorithm builds upon the Fourier mode decompositions to create a descriptive time-frequency analysis (TFA) framework that exhibits improved frequency resolution compared to traditional TFA techniques. The NFMD is similar in principle to one of the most common traditional techniques for TFA, the STFT. The STFT determines the frequency-domain energy spectrum of temporally local intervals of a signal. Although effective, the reliance on the traditional Fourier transform technique limits the frequency resolution of the STFT and has limited interpretability for understanding the signal components. Figure 4.1 presents a graphical comparison between the time-frequency analysis approach of the STFT and the NFMD algorithm. In the STFT, the original signal is multiplied by a set of windowing functions, such as a Gaussian, that is progressively applied to subsets of the signal. The signal subsets are then analyzed by the FFT. In the NFMD, signal segments are sliced and analyzed individually; the resultant Fourier modes are then fit by the FMD algorithm described in Section 4.2.1, thereby enabling a time-frequency analysis.

Specifically, NFMD subsamples segments of the time series of length ξ. These segments take the form $\boldsymbol{\chi}_i = \{\mathbf{z}(t_i), \mathbf{z}(t_{i+1})..., \mathbf{z}(t_{i+\xi/2})\}$. There are $n - \xi$ segments considered for an input signal of length n. The set of all segments in the signal is the set $\mathbf{X} = \{\boldsymbol{\chi}_1, \boldsymbol{\chi}_2, ..., \boldsymbol{\chi}_i, ..., \boldsymbol{\chi}_{n-\xi}\}$. The NFMD algorithm then applies the Fourier mode decomposition to each segment in $\mathbf{X}$. The NFMD algorithm is presented in Algorithm 4.

The NFMD algorithm takes in a set, $\mathbf{X}$, of window segments, $\boldsymbol{\chi}_i$, and for each $\boldsymbol{\chi}_i \in X$ learns a coefficient vector $\mathbf{A}_i$ and frequency vector $\boldsymbol{\omega}_i$ using FMD (Algorithm 3). Importantly, the learned frequency vector from each segment is used as the initial guess to the subsequent segment. This significantly increased the speed of the algorithm, by allowing each segment to start with a good initial guess for the frequency

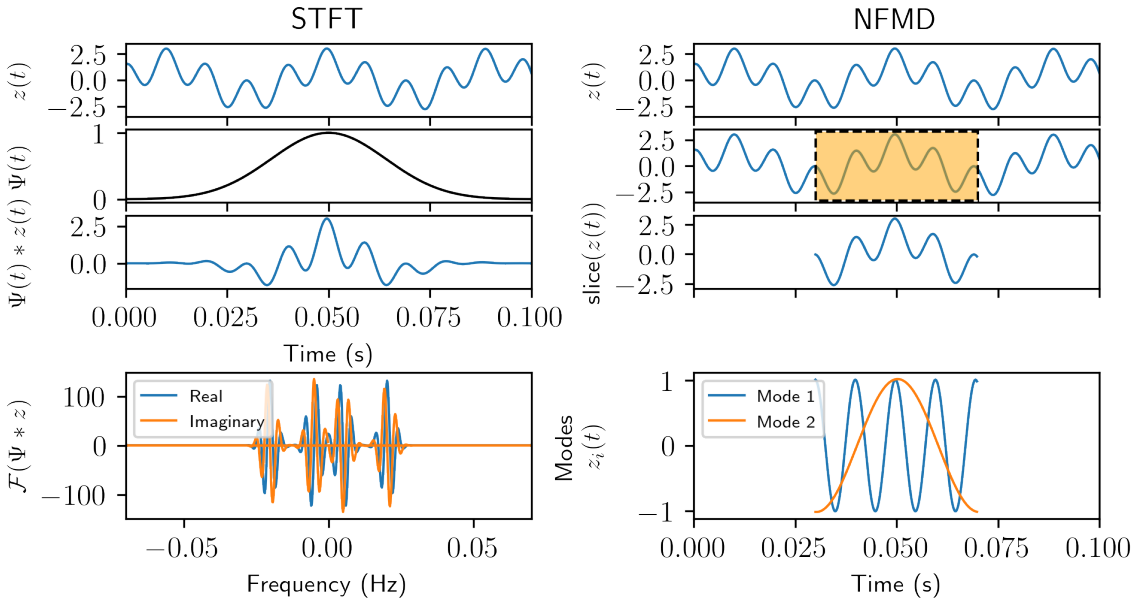


Figure 4.1: **Comparison between STFT and NFMD methods.** The top row shows an input signal, the second row shows a windowing function, and the third row shows the windowed signal. The fourth row provides the output of the decomposition. STFT computes the Fourier transform of a convolution of the windowed function, $\Psi(t)$. The example shown uses a Gaussian windowing function, which is commonly referred to as the Gabor transform. The bottom panel for STFT shows the FFT of the signal with real and imaginary components of the Fourier transform. The NFMD analyzes a segment of the signal, and fits a finite number of Fourier modes to the signal segment. The bottom panel for NFMD shows the two modes fit to the signal segment.

vector $\boldsymbol{\omega}_i$. The learned vectors from all of the segments are stored in matrices

$$\hat{\mathbf{A}} = \begin{bmatrix} - \mathbf{A}_1 - \\ \vdots \\ - \mathbf{A}_i - \\ \vdots \\ - \mathbf{A}_{n-\xi} - \end{bmatrix}, \tag{4.3}$$

Algorithm 4 Nonstationary Fourier Mode Decomposition

Input: Signal segments X, Window size ξ, tolerance tol

Output: Frequency matrix, $\hat{\omega}$, Coefficients matrix $\hat{\mathbf{A}}$, residual error vector $\hat{E}$

procedure
  $\boldsymbol{\omega}_{-1} \leftarrow \text{maxima}(\text{FFT}(\boldsymbol{\chi}_1))$ ▷ Initial guess for frequencies using the FFT
  for $\boldsymbol{\chi}_i \in X$ **do**
    $\mathbf{A}_i, \boldsymbol{\omega}_i, E_i = \text{FMD}(\boldsymbol{\chi}_i, \boldsymbol{\omega}_{-1}, tol)$ ▷ Learn Fourier modes for segment $\boldsymbol{\chi}_i$
    $\boldsymbol{\omega}_{-1} \leftarrow \boldsymbol{\omega}_i$
  end for
  $\hat{\mathbf{A}} = [\mathbf{A}_1^T, \cdots, \mathbf{A}_i^T, \cdots, \mathbf{A}_{n-\xi}^T]^T$ ▷ Coefficients matrix
  $\hat{\boldsymbol{\omega}} = [\boldsymbol{\omega}_1^T, \cdots, \boldsymbol{\omega}_i^T, \cdots, \boldsymbol{\omega}_{n-\xi}^T]^T$ ▷ Frequencies matrix
  $\hat{\mathbf{E}} = [E_1, \cdots, E_i, \cdots, E_{n-\xi}]$ ▷ Residual errors vector
  return $\hat{\mathbf{A}}$, $\hat{\boldsymbol{\omega}}$, $\hat{\mathbf{E}}$
end procedure

and

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} - \boldsymbol{\omega}_1 - \\ \vdots \\ - \boldsymbol{\omega}_i - \\ \vdots \\ - \boldsymbol{\omega}_{n-\xi} - \end{bmatrix}. \tag{4.4}$$

The learned frequency vectors, $\boldsymbol{\omega}_i = [\omega_{1,i}, ..., \omega_{k,i}, ...\omega_{K,i}]$, and coefficient vectors, $\mathbf{A}_i = [A_{1,i}, ..., A_{k,i}, ..., A_{K,i}]$, contain the frequencies and coefficients for the K Fourier modes of each signal segment $\boldsymbol{\chi}_i \in X$.

4.2.3 Instantaneous Frequency and Amplitude

The NFMD decomposes nonstationary, nonlinear signals by fitting a set of Fourier modes to a set $\mathbf{X}$ of signal segments, as previously described. Importantly, for each

signal segment $\boldsymbol{\chi}_i$ the algorithm yields a frequencies vector $\boldsymbol{\omega}_i$ and coefficients vector $\mathbf{A}_i$. For each Fourier mode k, a vector of instantaneous frequencies can be constructed by collecting the learned Fourier mode frequency $\omega_{k,i}$ for each of the signal segments χ_i for a given mode k. The instantaneous frequency vector takes the form

$$\boldsymbol{\omega}_k = [\omega_{k,1}, ..., \omega_{k,i}, ..., \omega_{k,n-\xi}].$$

The instantaneous amplitude of a Fourier mode can be computed from its coefficients $a_{k,i}$ and $b_{k,i}$ with the relationship

$$\phi_{k,i} = \sqrt{(a_{k,i})^2 + (b_{k,i})^2}.$$

Similar to the instantaneous frequency vector, an instantaneous amplitude vector can be constructed for each mode k with one element corresponding to each segment $\boldsymbol{\chi}$,

$$\boldsymbol{\phi}_k = [\phi_{k,1}, ..., \phi_{k,i}, ..., \phi_{k,n-\xi}].$$

These metrics are useful for comparing the time-frequency analysis from NFMD to other existing time-frequency analysis methods that report instantaneous amplitude and instantaneous frequency of an input time series signal.

4.2.4 Nonstationary Signals

NFMD can provide insight into nonstationary signals of the form

$$z(t) = \mu(t) + \sum_{l=1}^{L} A_l(t) \exp(i\phi_l(t)), \tag{4.5}$$

where $\mu(t)$ is an unknown, non-periodic function describing the nonstationary part of the signal and there are L periodic modes with the amplitude function $A_l(t)$ and instantaneous phase function $\phi_l(t)$. The term $\mu(t)$ is only assumed to be continuous. We will call the function $\mu(t)$ the signal's *instantaneous mean*. The concept of an instantaneous mean will prove useful in the analysis of nonstationary signals where the signal mean drifts or trends away from zero.

The frequency and coefficient vectors allow computation of individual Fourier modes $F_{k,i} = a_{k,i}\cos(\omega_{k,i}t) + b_{k,i}\cos(\omega_{k,i}t)$ for each of the modes $k \in [1, K]$ and signal segments $\boldsymbol{\chi}_i$. In signals that exhibit a moving mean, there is a mode that will account for the nonstationary part of the signal. The instantaneous mean of the signal, $\mu(t)$, can be estimated from the Fourier modes $F_{k,i}$ corresponding to the mode that accounts for the nonstationary part of the signal. Many interpolation strategies can be implemented to compute the instantaneous mean. We implement a simple strategy of concatenating the value of the Fourier mode F_k at the median time from each time segment $\boldsymbol{\chi}_i$. To visualize this implementation, imagine we construct the matrix

$$\mathcal{M} = \begin{bmatrix} - F_{k,1} - \\ \vdots \\ - F_{k,i} - \\ \vdots \\ - F_{k,n-\xi} - \end{bmatrix} = \begin{bmatrix} F_{k,1}(t_1) & \dots & F_{k,1}(t_{1+\xi/2}) & \dots & F_{k,1}(t_{1+\xi}) \\ \vdots & & \vdots & & \vdots \\ F_{k,i}(t_i) & \dots & F_{k,i}(t_{i+\xi/2}) & \dots & F_{k,i}(t_{i+\xi}) \\ \vdots & & \vdots & & \vdots \\ F_{k,n-\xi}(t_{n-\xi}) & \dots & F_{k,n-\xi}(t_{n-\xi/2}) & \dots & F_{k,n-\xi}(t_n) \end{bmatrix}.$$

The approach is to use the center column of the matrix $\mathcal{M}$ as the instantaneous mean. This mean is defined as

$$\boldsymbol{\mu} = [F_{k,1}(t_{1+\xi/2}), ..., F_{k,i}(t_{i+\xi/2}), ..., F_{k,n-\xi}(t_{n-\xi/2})].$$

The mode representing the mean can be challenging to identify, and generally requires inspection of each of the modes identified by NFMD. In this work, the instantaneous mean is always the lowest-frequency mode.

4.3 Results

We benchmark the NFMD against the Hilbert-Huang Transform for time-frequency analysis of a series of nonstationary multi-component signals. The two methods are compared with and without noise, and on signals with abrupt changes in instantaneous frequency and instantaneous amplitude. After comparing NFMD with the HHT, a

pair of oscillator examples are provided to demonstrate how the NFMD and the discovered instantaneous mean can provide insight into the forcing function applied to the oscillator. Finally, the method is applied to simulated and experimental data for a real-world oscillator-based microscopy method. The instantaneous mean is proven to be effective in an experimental data set by validating the form of the discovered instantaneous mean against experimental control data with known forcing functions.

4.3.1 Synthetic Test Signals

Consider a basic multi-component signal with a non-periodic mean:

$$\begin{aligned} z(t) &= z_1(t) + z_2(t) + \mu(t) \\ z_1(t) &= A_1(t)\cos(2\pi\omega_1(t)t) \\ z_2(t) &= A_2(t)\cos(2\pi\omega_2(t)t), \end{aligned}$$

with amplitude, phase, and instantaneous mean functions

$$\begin{aligned} A_1(t) &= 1 + 0.5\exp(-t/3) \\ \omega_1(t) &= 360 - 10\exp(-t/0.5) \\ A_2(t) &= 8 - 0.5\exp(-t) \\ \omega_2(t) &= 80 - 2t \\ \mu(t) &= 1.5 + 2.5\exp(-x/1.5), \end{aligned}$$

where $z(t)$ is the signal, $z_1(t)$ and $z_2(t)$ are the periodic components in $z(t)$, and $\mu(t)$ is the slow-moving non-periodic mean. Figure 4.2 shows a traditional STFT spectrogram and PSD analysis of the signal, and Figure 4.3 shows the true periodic modes, with instantaneous parameters, and the mean $\mu(t)$. The signal is generated over one second with sampling interval $\Delta t = 2 \times 10^{-4}$ s.

Signal decomposition is performed with both the NFMD and HHT. The HHT employs the EMD to identify a set of empirical modes. The EMD takes a number of

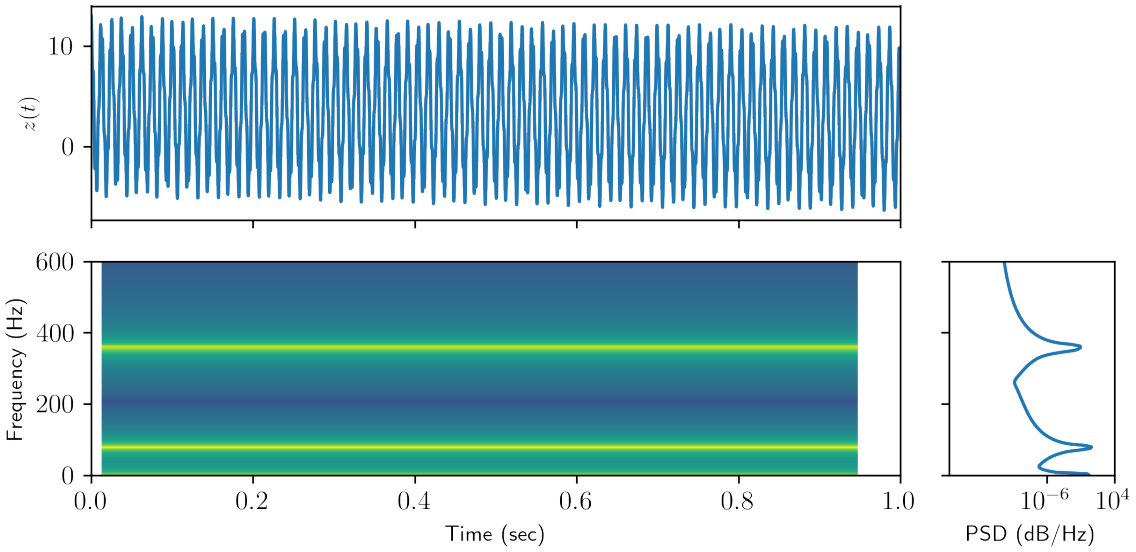


Figure 4.2: **Example signal for demonstrating NFMD.** The true signal is presented with the short time Fourier transform and the power spectral density (PSD). The signal has three maxima that appear in the PSD and STFT, indicating that there are likely three modes to be considered.

hyperparameters which adjust the outputs, including the number of identified modes. The HHT used in this work used $\theta_1 = 0.05$, $\theta_2 = 0.5$, $\alpha = 0.05$, and identified four modes in the signal. The two most important hyperparameters for the NFMD are the window size (similar to STFT) and the number of modes to fit to the data. For this data, three modes are fit to a window size of 250 points (or 0.05 seconds). Figure 4.4 shows the decomposed signal for both HHT and NFMD, where NFMD does a significantly better job of identifying both the instantaneous frequency and instantaneous amplitude of the signal. Adding noise to the signal significantly affects the HHT's decomposition when compared to the NFMD, as demonstrated in Figure 4.5. Noise in the signal is added at a signal-to-noise ratio (SNR) of 35. The SNR in this work is defined as $\text{SNR} = 10\log_{10}(\|u(t)\|_2^2/\|(\tilde{u}(t) - u(t)\|_2^2)$.

NFMD enables signal decomposition even for noisy signals where the HHT begins to have difficulty identifying intrinsic mode functions. The NFMD propagates the

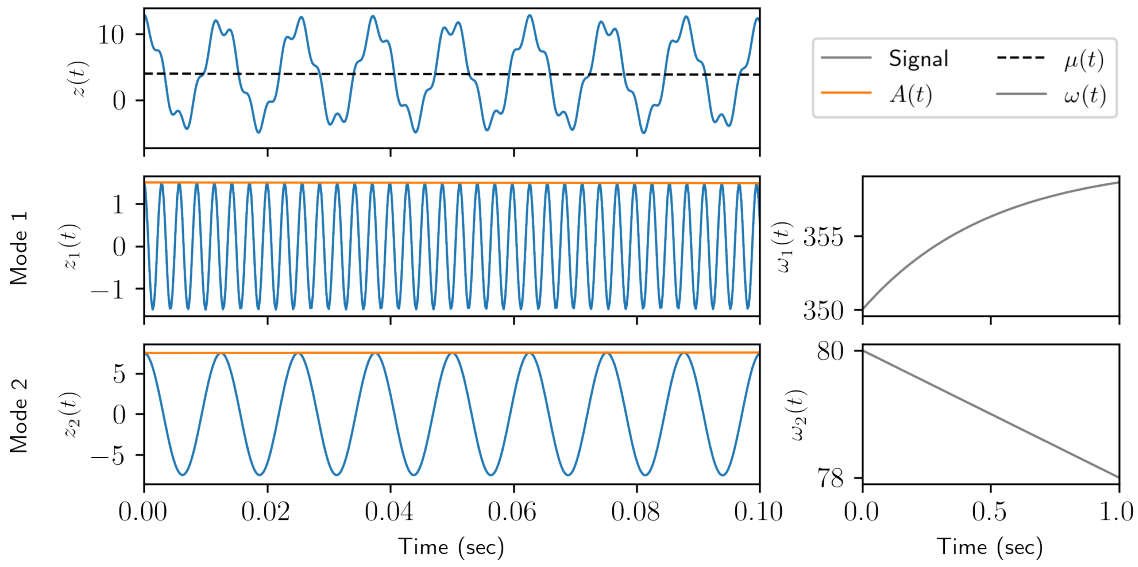


Figure 4.3: **Instantaneous mean, amplitude, and frequency** of the example signal. Only part of the signal, from $t = 0$ to $t = 0.1$ is shown. The instantaneous frequency, $\omega(t)$, and amplitude, $A(t)$, are presented for the two periodic modes. The mean, $\mu(t)$ of the signal is denoted on the signal itself.

noise from the signal through the decomposition process, showing some noise-like error in the estimated instantaneous frequency vectors, amplitude vectors, and mean. However, the NFMD provides clearly superior decomposition of the signal. It is worth noting the NFMD makes an error early in its estimation of the first periodic mode in Figures 4.4 and 4.5. The error simultaneously underestimates the amplitude and overestimates the frequency of the first mode. We attribute this edge effect to applying the algorithm to an incomplete period in the initial time step which is corrected in subsequent time steps.

One particular advantage of NFMD is the ability to correctly identify sharply-changing instantaneous frequency in periodic modes or abrupt changes in the instantaneous mean of an input signal. This is an advantage against other modern optimization-based methods built on VMD, because VMD prioritizes finding solutions with smooth amplitude functions [95, 86]. However, the NFMD has no preference for

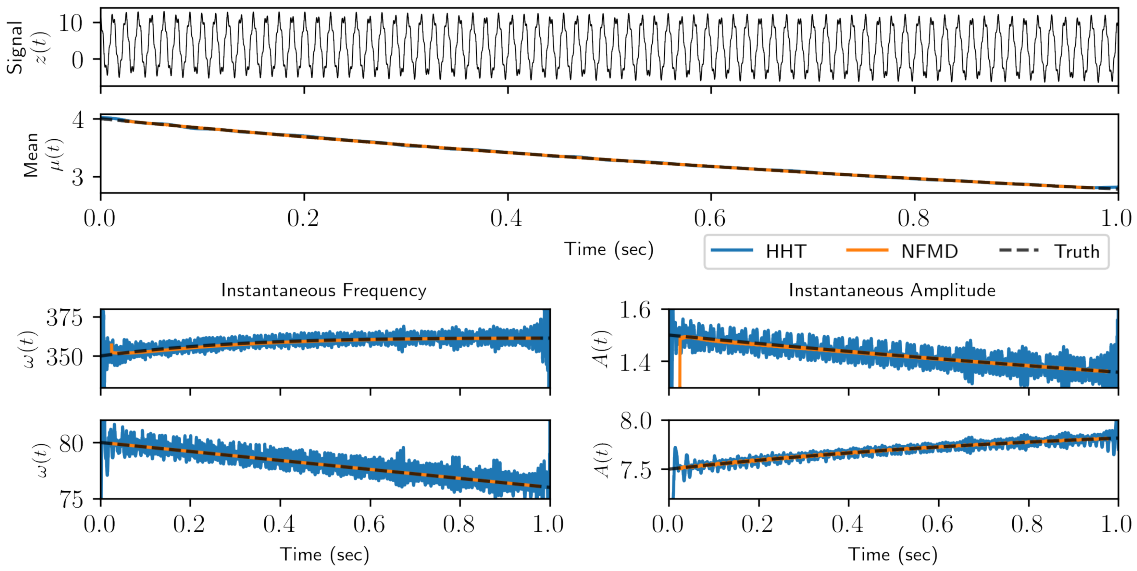


Figure 4.4: **Decomposition of example signal by NFMD and HHT without noise.** The estimated instantaneous mean mode is shown for both the HHT and NFMD decomposition methods. The instantaneous frequency and amplitude of the two periodic components is also shown for both decomposition approaches.

smooth amplitudes and can accurately decompose these types of signals with abrupt changes in the instantaneous frequency, instantaneous amplitude, or mean. For example, consider a signal with an abrupt change in the instantaneous frequency of one of its modes at $t = 0.5$. The model for this example is

$$z(t) = z_1(t) + z_2(t) + \mu(t)$$
$$z_1(t) = A_1(t)\cos(2\pi\omega_1(t)t)$$
$$z_2(t) = A_2(t)\cos(2\pi\omega_2(t)t),$$

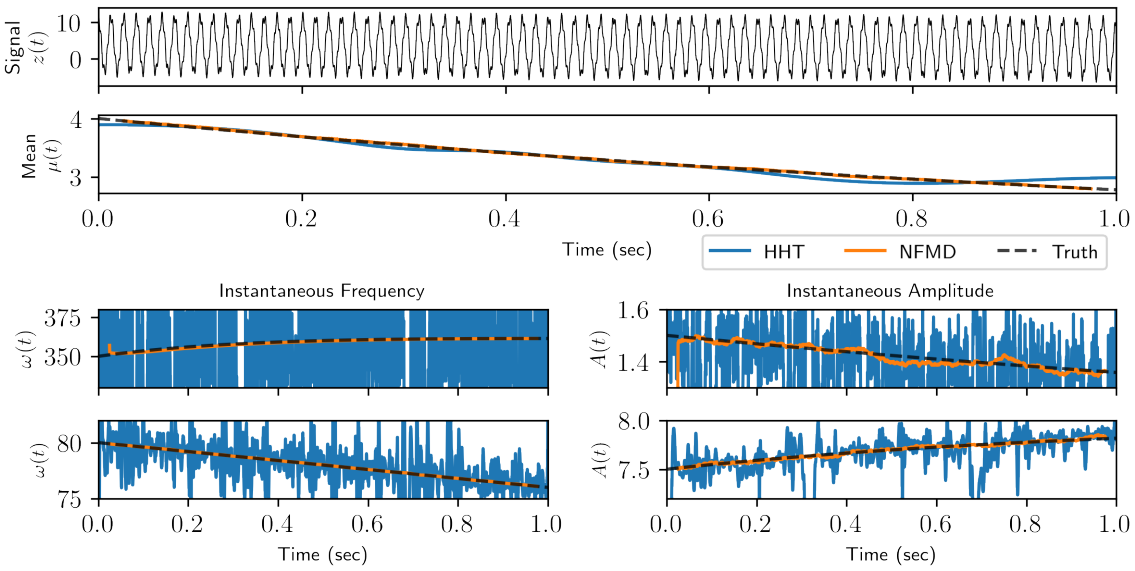


Figure 4.5: **Decomposition of example signal with added noise (SNR=35).** The NFMD correctly identifies the instantaneous frequency, amplitude, and mean of the signal, although it does propagate noise in the system. The HHT begins to show erratic behavior in its estimates of all instantaneous parameters.

with amplitude, phase, and instantaneous mean functions

$$
\begin{aligned}
A_1(t) &= 2 + \exp(-t/4) \\
\omega_1(t) &= 400 + 10H(0.5)(1 - \exp((t - 0.5)/0.1) \\
A_2(t) &= 2t + 2 \\
\omega_2(t) &= 60 - t \\
\mu(t) &= 1.5 + 2.5\exp(-x/1.5),
\end{aligned}
$$

where $H(0.5)$ is the Heaviside function centered at $t = 0.5$s. Noise is added to the signal at an SNR ratio of 25 and decomposed by both HHT and NFMD. The decomposed signal is shown in Figure 4.6. The NFMD correctly identifies the sharp transition in frequency, and consequently obtains qualitatively good estimates on the amplitude of both periodic modes. Results from the HHT are presented, but it is important to note that in cases like this example, the HHT often yields extra intrinsic

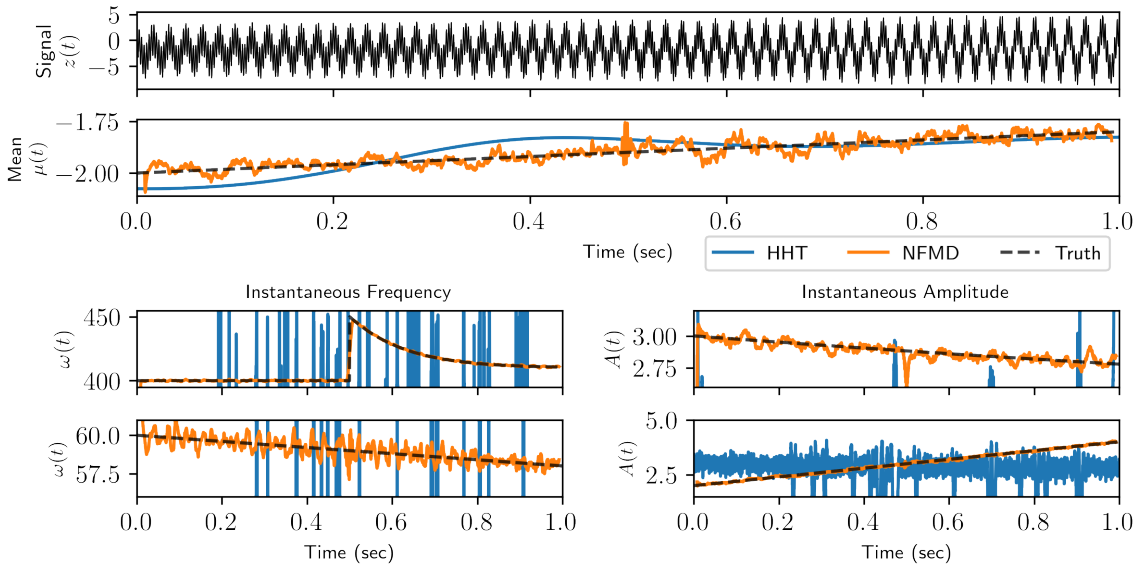


Figure 4.6: **Decomposition of signal with discontinuity in instantaneous frequency of a periodic mode.** Data contains additive white noise with SNR=25. The NFMD correctly identifies the three modes, including the instantaneous mean and the frequencies and amplitudes of the periodic modes. The HHT fails to identify the correct number of modes and assigns extra modes to the data. The periodic modes uncovered by HHT do not qualitatively match any of the correct signal modes.

mode functions. It is challenging to assign which of the HHT modes are intended to represent which periodi‘c mode, so the presented results were quantitatively closest to the 'true' mode by comparing both instantaneous frequency and instantaneous amplitude vectors. Additionally, all of the remaining modes from HHT are summed together to estimate the instantaneous mean. Although the HHT yields a reasonable looking instantaneous mean, neither of the periodic modes are correctly decomposed in this example.

As another example, we consider a situation where the instantaneous mean of the signal changes abruptly in the middle of the signal ($t = 0.5$ seconds). The input signal

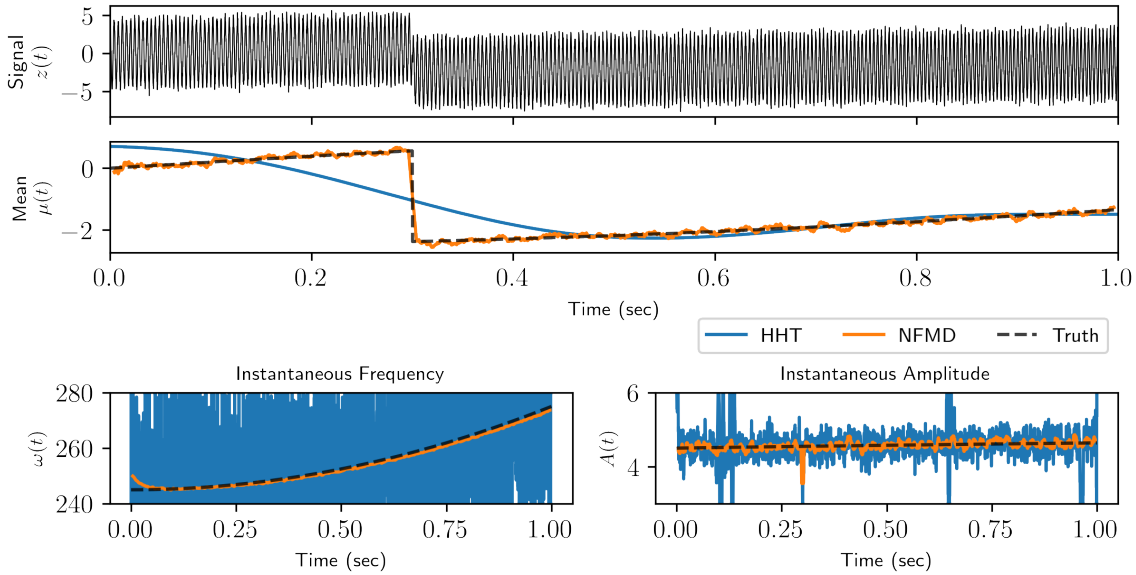


Figure 4.7: **Decomposition of signal with discontinuity in instantaneous mean.** Data contains additive white noise with SNR=20. The NFMD correctly identifies the periodic mode instantaneous frequency and amplitude, and the instantaneous mean of the signal. The HHT fails to identify either the mean or the periodic component.

uses the model

$$z(t) = z_1(t) + \mu(t)$$
$$z_1(t) = A_1(t)\cos(2\pi\omega_1(t)t)$$

with amplitude, phase, and instantaneous mean functions

$$A_1(t) = 5 - 0.5\exp(-t/3)$$
$$\omega_1(t) = 245 + 10t^2$$
$$\mu(t) = \begin{cases} \sin(2t) & t \leq 0.25 \\ -2.5\cos(t) & t > 0.25. \end{cases}$$

The signal has added Gaussian noise with SNR= 20. As shown in Figure 4.7, the NFMD accurately estimates instantaneous signal mean and identifies the correct periodic mode. The HHT fails to find the instantaneous frequency of the periodic mode,

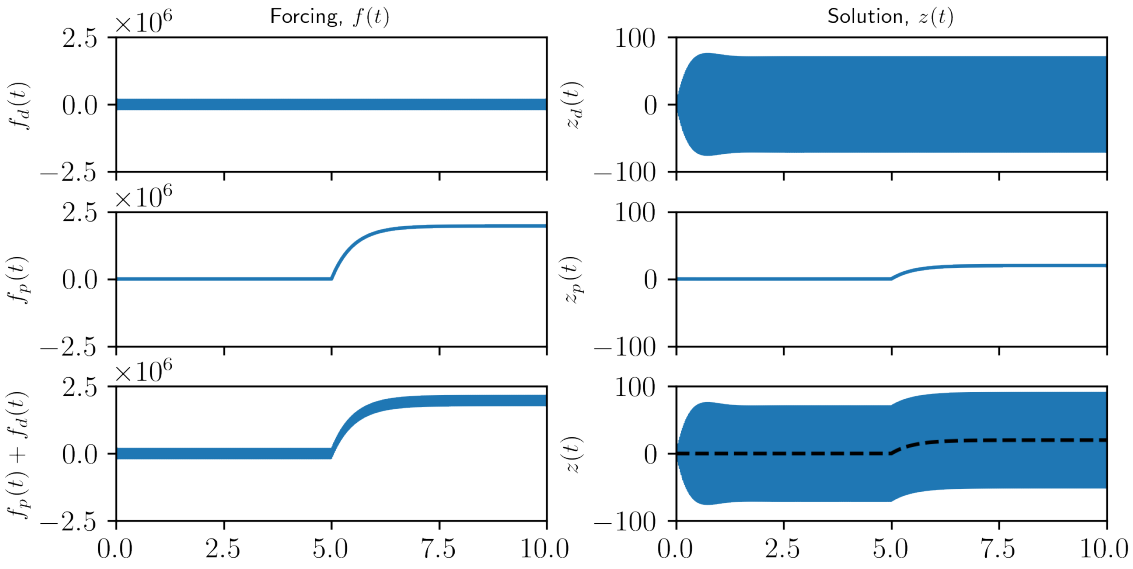


Figure 4.8: **Harmonic oscillator solutions with various forcings.** The left column shows the applied forcing, and the right column shows the solution to the harmonic oscillator assuming it starts from rest. The top row shows a periodic driving force, the middle row shows a non-periodic perturbation, and the bottom row shows a superposition of the two forcings.

and errantly suggests a low-frequency wave as the instantaneous mean.

Having demonstrated that NFMD offers material advantages over HHT methods for TFA, both in terms of mode extraction in noisy signals and for reacting to sharp changes in instantaneous parameters, we next discuss applications to realistic systems.

4.3.2 Applications

The example synthetic test signals above illustrate the power of NFMD to decompose signals with abruptly changing instantaneous parameters and accurately estimate the instantaneous mean of noisy signals. One application of the instantaneous mean is estimating the non-periodic forces applied to an oscillatory system. Consider the ubiquitous harmonic oscillator

$$\ddot{x} + 2\beta\omega_0\dot{x} + \omega_0^2 = F(t)/m,$$

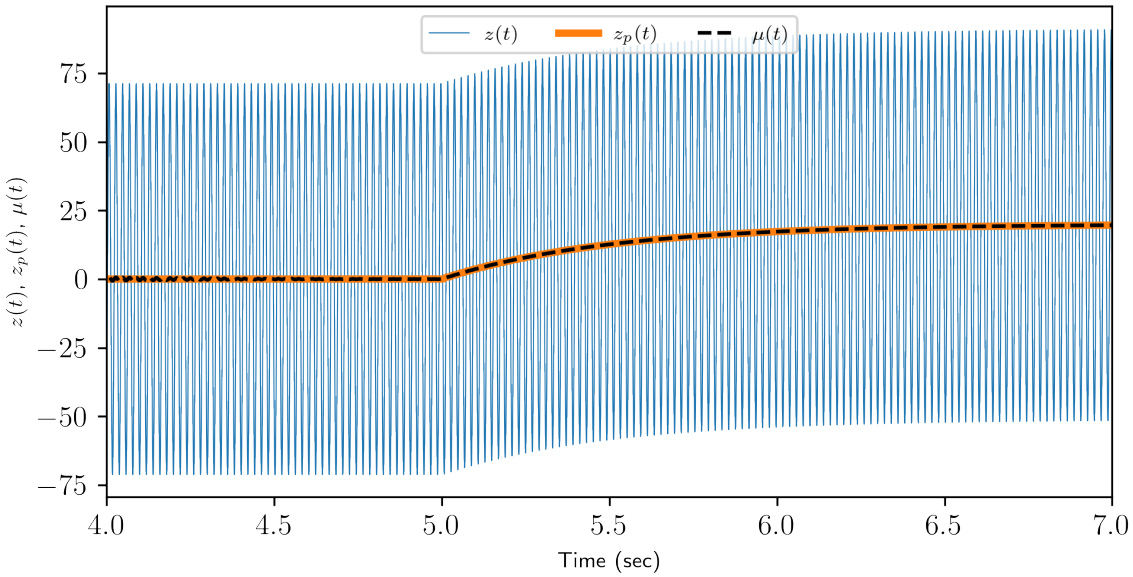


Figure 4.9: **The signal mean (dashed) of the perturbed, periodically driven oscillator matches the solution of the perturbation-only oscillator (orange).** This enables NFMD to directly probe non-oscillatory driving forces applied to oscillators by accurately recovering the signal mean of a nonstationary signal.

where x is the position of the oscillator, β is a damping factor, ω_0 is the resonant frequency of the oscillator, $F(t)$ is an applied forcing, and m is the mass of the oscillator. We consider the solution to this oscillator for three types of applied forcing functions: a periodic driving force, $F_d(t)$, and a non-periodic perturbation forcing, $F_p(t)$, and a combination of periodic and perturbation forcing functions, $F_d(t)+F_p(t)$.

The driving force $F_d(t)$ is of the form $\alpha e^{i\omega t}$, where ω is the frequency and α is the amplitude of the driving force. The perturbation forcing function has the general form

$$F_p(t) = +H(t-t')\gamma(1-e^{-(t-t')/\tau}), \tag{4.6}$$

where H is the Heaviside function, t' is a perturbation onset time, and τ is a characteristic relaxation time constant. An oscillator with this forcing function will have a solution of the form

$$x_p(t) \propto \phi e^{i\omega t} + \psi e^{-(t-t')/\tau}, \tag{4.7}$$

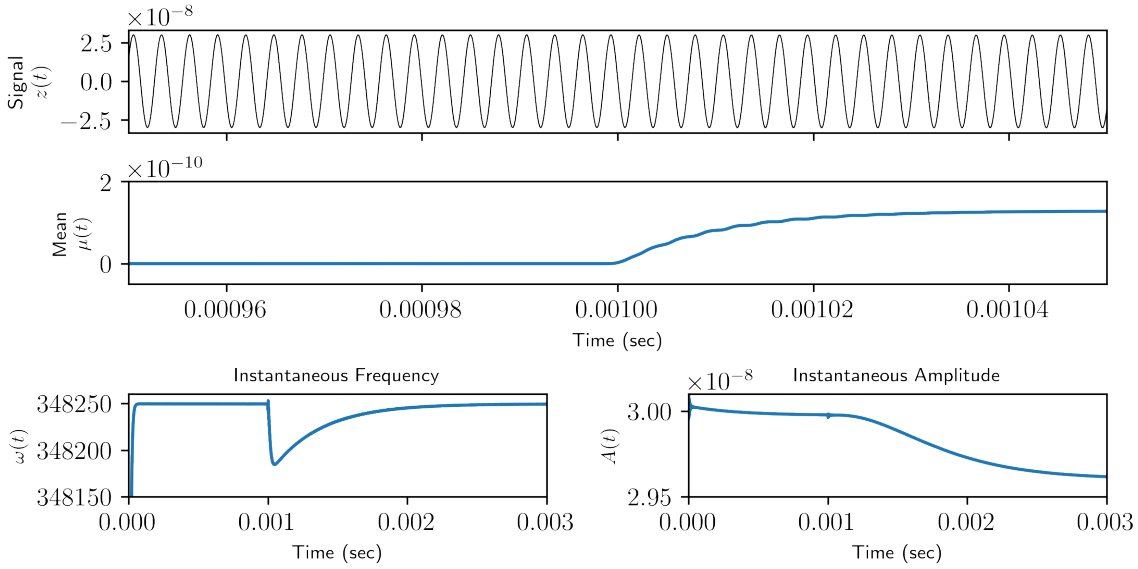


Figure 4.10: **Decomposition of forced oscillator signal by NFMD.** The NFMD decomposes a harmonic oscillator forced with a periodic forcing function and non-periodic perturbation. The instantaneous mean of the signal, $\mu(t)$, is directly correlated to the non-periodic forcing.

where ϕ is a prefactor determined by the parameters in the governing equation. Forcing functions and solutions for an oscillator forced by $F_d(t)$, $F_p(t)$, and $F_p(t) + F_d(t)$ are presented in Figure 4.8.

The solution to the combined case ($F(t) = F_d(t) + F_p(t)$) has an instantaneous mean that is nearly identical to the solution for the perturbation-only ($F(t) = F_p(t)$) case. This enables the NFMD signal decomposition to provide insight into the form of the non-periodic forcing applied to an oscillator by estimating the instantaneous mean of the signal. The instantaneous mean and the perturbed solution are plotted on top of the solution to the driven, perturbed solution in Figure 4.9.

We confirm this approach works by using a series of numerical simulations. The same periodic driving force, $F_d(t)$, is used for all simulations while a set of different perturbation forces, $F_p(t)$, is applied to each oscillator. The perturbation forces have different relaxation times, τ in equation 4.6. The relaxation times vary from 10^{-7} to

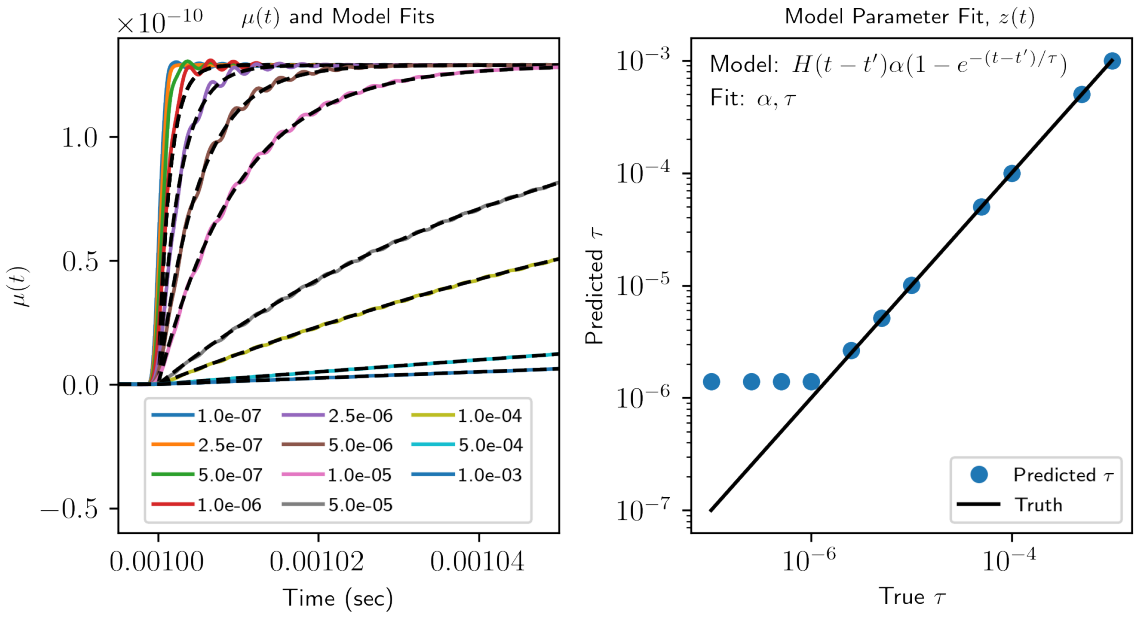


Figure 4.11: **Model fit comparison between instantaneous mean from NFMD and true parameters from non-periodic forcing.** The simulated oscillator is subjected to a perturbation at $t = 0.001$ seconds (left panel). A model is fit to the discovered instantaneous mean mode $\mu(t)$. The time constant τ in the fit model is compared to the time constant of the perturbation force $F_p(t)$ (right panel).

10^{-3} s. The simulated oscillators are all subjected to the combined driving force and perturbation force. The signals have added white noise with SNR= 100.

Figure 4.10 shows the decomposition of a single simulated oscillator. Note the time-varying mean of the signal correlates directly to the non-periodic forcing function. Figure 4.11 shows the signal means discovered with the NFMD and models fit to the instantaneous means. The model $H(t-t')\alpha(1-\exp((t-t')/\tau)$ is fit to each of the instantaneous means. This model makes it possible to identify the relaxation time τ in the perturbation function. Below approximately $1\mu s$, the estimated τ begins to flatten out. This is a result of the window size (ξ) used for model fitting, which is approximately a $1\mu s$ window width.

In this proposed harmonic oscillator application, the NFMD can identify the in-

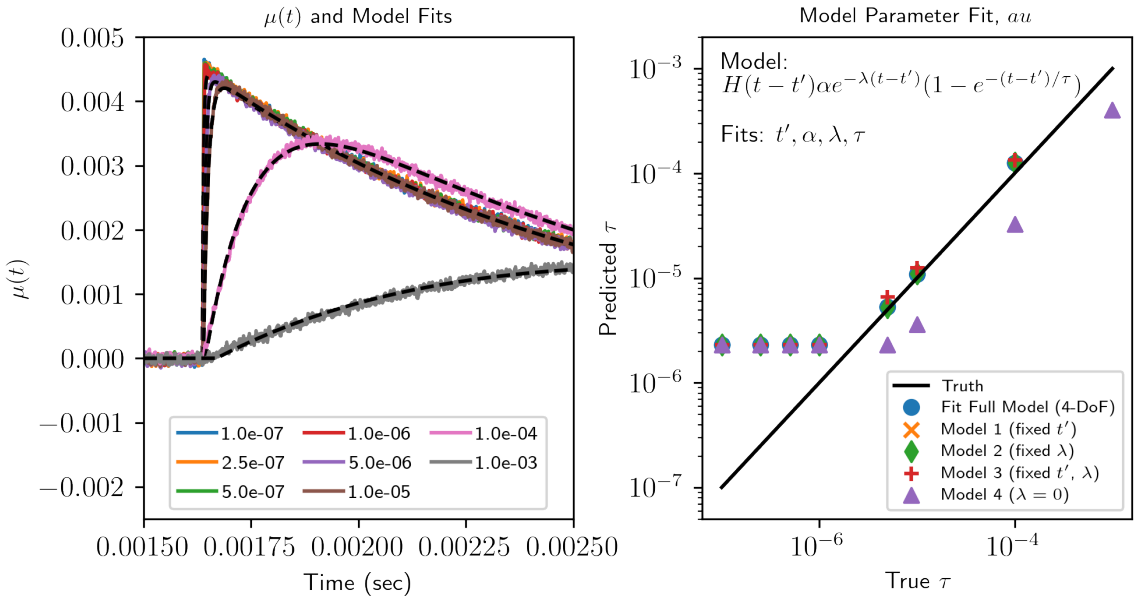


Figure 4.12: **NFMD decomposition of experimental trEFM control data with controlled perturbation time constants.** A model is fit which contains the time constant τ and compared with truth values.

stantaneous mean of a signal. The instantaneous mean of the signal is then fit to a model, which can provide insight to the form of the non-periodic perturbation applied to the system.

4.3.3 Experimental Application

We test this application on a real-world cantilever based imaging modality termed time-resolved electrostatic force microscopy (trEFM), which is an electrical modification applied to the nanoscale imaging technique of atomic force microscopy [118, 103, 119, 101, 120]. In trEFM, a periodically driven metallic cantilever is brought into close proximity to a surface. An electric field generated by an accumulation of electrical charge on the surface imparts a force on the metallic cantilever, as well as an electrostatic force gradient [121]. The electrostatic force is typically triggered by an external stimulus, such as a voltage signal or photogenerated charge via an optical

excitation source. Such methods are useful for extracting dynamic information in nanoscale measurements of photovoltaic or ionic conducting systems via the effect of the force on the cantilever's resonance frequency. In a typical trEFM experiment, the desired outcome is extracting an unknown characteristic time constant τ, usually describing the time-dependent change in the electrostatic force gradient [118, 122, 114, 123]. As a test case for NFMD, we apply a series of perturbation forces to the cantilever of the form (4.6). In this experiment, the relaxation time τ is controlled in the applied forcing, providing a set of experimental data with different, known relaxation times.

The NFMD is used to decompose the experimentally-measured cantilever signal into a single periodic mode (with instantaneous frequency near the periodic driving force) and an instantaneous mean. The instantaneous mean can then be fit to a model for the forcing term. The model we used is

$$\mu(t) = H(t-t')\alpha \exp\left(-\lambda(t-t')\right)(1-\exp\left((t-t')/\tau\right),$$

where $H(t-t')$ is the Heaviside function centered at $t-t'$, t is the time, t' is the perturbation onset time, α is an amplitude constant, λ is a decay constant, and τ is the relaxation time. The term with the decay constant λ was included in this model to fit a recurring pattern in the data that is likely a constant related to the cantilever. Five different versions of this model were fit to the data: one with a fixed perturbation time $t' = 0.168\mu$s, one with a fixed decay constant $\lambda = 1080$, one with both parameters fixed, one with λ set to zero, and one with all parameters fit by the model. Figure 4.12 shows the result of fitting these models to the instantaneous means of the experimental data.

Most of the models perform similarly, though the model with the decay constant λ set to zero tends to underestimate the correct relaxation time. Similar to the simulation results, the same trend occurs around $\tau = 1\mu s$, where the model no longer fits the truth line and the predicted τ flattens out. Unfortunately, the size of the

window is limited by the frequency of the periodic components of the input signal. In both cases, the 1μs window was the minimum window size where the NFMD decomposition successfully decomposes the signal.

It is important to compare this approach with the existing methods for trEFM time constant estimation [120]. One current method is using the instantaneous frequency vector [124], $\omega_k(t)$, of the periodic mode and estimating the time between the perturbation onset and the next local minima in the instantaneous frequency curve. Prior research showed an empirical correlation between this time interval and the relaxation time of the perturbation [101, 124]. To define the particular experiment, a calibration curve is used to estimate a relaxation time given the instantaneous frequency vector. This approach is effective, and enables identification of sub-microsecond relaxation time constant [101]. However, the main drawback of the method in [124] is that the correlation is indirect. Therefore, the external perturbation (namely, τ) is not directly learned, and the calibration curve will change based on experimental variables such as the cantilever being used with a different set of physical parameters like quality factor and spring constant. For more complicated systems where the relevant timescales are more than single exponential (modern photovoltaic systems with ionic transport and dielectric relaxation in battery materials), the lack of a defined model in this calibration curve can prove limiting [125, 114]. A chief advantage of the proposed NFMD approach is that the forcing function can be detected directly from experimental data via the instantaneous mean.

4.4 Conclusion

Time-frequency analysis methods are critically important in science and engineering. In this work, we develop a data-driven approach to time-frequency analysis that helps address shortcomings of classic approaches, including the extraction of nonstationary signals with discontinuities in their behavior. By integrating elements of modern gradient descent algorithms, the Fourier transform, multi-resolution analysis, and

Bayesian spectral analysis, we can learn an interpretable Fourier mode-based model for analyzing nonstationary signals with periodic components, thus circumventing the deleterious effects normally associated with nonstationary processes and allowing for accurate identification of instantaneous frequencies and their amplitudes. Indeed, our method is equivalent to a *nonstationary Fourier mode decomposition* (NFMD) for nonstationary and nonlinear temporal signals. Importantly, it produces interpretable signal decompositions that can handle signals with multiple periodic components, nonlinear phase functions, and sharp discontinuities in the phase function or periodic mode amplitudes. The method results in a superior time-frequency analysis to the HHT for nonstationary signals, and improves both temporal and spatial resolutions compared to the STFT, thus providing a viable and broadly applicable architecture for integration into a diverse number of scientific processes.

Chapter 5

CONCLUSION

This thesis presents three innovations in data-driven modeling relevant to the domain of materials science and engineering. First, a technique for data-driven modelings of boundary value problems was developed. The method can identify the differential operator governing boundary value problems, including spatially varying parametric coefficients in the operator. It builds on previous works in systems identification and sparse regression to identify a parsimonious, data-driven model of the system. Next, we present a data-driven approach for linearizing nonlinear boundary value problems enabled by deep neural network autoencoders. Autoencoders transform the nonlinear problem to a latent space where linear properties hold, which allows the discovery of an invertible linear operator that governs the system. Finally, we describe a time-frequency analysis algorithm powered by gradient descent and Bayesian spectral analysis that enables decomposition of nonstationary multimodal time series signals. The algorithm uses a sliding window to analyze segments of the signal, enabling a high-resolution time-frequency analysis superior to traditional techniques.

Sparse identification of nonlinear and linear operators in boundary value problems

Chapter 2 presents an algorithm for identifying the nonlinear (or linear) differential operator of a boundary value problem via sparse regression. It is compatible with systems containing spatial heterogeneity and depends on spatial measurements of steady-state, time-invariant systems. The method extends existing literature that

uses sparse regression for system identification, and specifically extends literature regarding the Sparse Identification of Nonlinear Dynamics (SINDy) lineage [14, 126, 25, 26]. It is the first algorithm in the SINDy family to be applied to time invariant problems, and established an approach for identifying spatially-varying coefficients and parsimonious operators simultaneously. We demonstrated the approach on four model systems, including those governed by the Sturm-Liouville operator, the Laplace operator, and the biharmonic operator commonly seen in the Euler-Bernoulli beam equation. The method was shown to have some resilience to noise, though fails to succeed in high-noise environments. The algorithm relies on a group regression strategy and a variety of different trials, where each trial presents a unique system configuration governed by the operator and the applied forcing functions. Each trial uses a different forcing function to get a different system configuration.

Discovery of linearizing transforms for nonlinear boundary value problems

Chapter 3 develops a deep learning method that utilizes a dual-autoencoder architecture to linearize nonlinear boundary value problems. The method depends on neural networks, which are commonly known as excellent generic function approximators, to learn a pair of linearizing transforms to project a nonlinear problem into a linear space. The linear space occurs in the latent space of the autoencoders. The method leverages the linear space to learn a linear invertible operator, enabling the use of the Green's function solution approach to compute new solutions to the system. The method is most closely related to the single-autoencoder architecture designed for dynamical nonlinear initial value problems [51]. The method in Chapter 3 was demonstrated on nonlinear one-dimensional Sturm-Liouville and biharmonic systems and a two-dimensional Poisson's equation. In every case, the learned operator and Green's function enabled prediction of the solution of new systems given a forcing function, even with forcing functions in different functional families than those used

for training (e.g. periodic and gaussian forcings in training but polynomials in testing). Furthermore, it was proven to be compatible with systems with spatially-varying parametric coefficients.

Time-frequency analysis of nonstationary time series data

Chapter 4 introduces a novel time-frequency analysis algorithm that enables high resolution in both frequency and time for nonstationary multimodal signals (i.e. signals with non-zero and changing mean and multiple oscillatory modes). The method combines the short-time Fourier transform, Bayesian spectral analysis, modern gradient descent algorithms, and multiresolution analysis to fit a finite number of Fourier modes to the data. The fit Fourier modes can have arbitrary frequency precision, unlike the discrete Fourier transform, and leverages Bayesian spectral analysis methods for estimating the amplitude of each Fourier mode. The algorithm identifies instantaneous frequency and amplitude of the oscillatory modes present in the data *and* can estimate the instantaneous mean of the signal. The algorithm extends a previous algorithm aimed at fitting Fourier modes to an entire data set, which enables high-resolution frequency estimation but does not provide a time-frequency analysis [75]. We prove the method works on a variety of example systems including noisy systems and real experimental data. The decomposition provides new insights for an experimental data set and enables discovery of the form of the forcing function applied to the experimental system.

Future directions

Sparse identification of nonlinear and linear operators in boundary value problems

The "SINDy-BVP" method explored the application of the SINDy algorithm to time-invariant boundary value problems. It developed a new way for discovering spatially-varying parametric coefficients through use of different trials. However, the biggest

drawback to the algorithm is how it fails in systems with high measurement noise. The noise problem might be addressed in a variety of ways including denoising procedures, such as filtering procedures or neural networks for estimating "true" system data given noisy system data. Alternative approaches that may indirectly help with the problems arising from noise include using a weak formulations of SINDy and imparting additional constraints on the spatially varying coefficients. Weak formulations of SINDy should reduce the effects of noise on the regression by convolution of the noisy data with clean test functions. Imparting additional constraints, such as regularizing against rapidly changing coefficient values, may also reduce the effects of noise and improve discovery of smooth coefficient vectors in noisy data.

Discovery of linearizing transforms for nonlinear boundary value problems

The DeepGreen method in Chapter 3 utilizes deep neural network autoencoders for learning a transformation that linearizes nonlinear boundary value problems. Although the method shows great utility and enables prediction of solutions for new system conditions, it fails to find unique linear representations of the system and yields different linear operators each time the networks are trained. This makes interpretation of the discovered operator and Green's function challenging. Future work should aim to place additional constraints on the learned operator and Green's function, and ideally learn unique operators for a given system. Additionally, it may be interesting to study how the architecture responds to nonlinear systems with non-unique solutions.

Time-frequency analysis of nonstationary time series data

The time-frequency analysis algorithm shown in Chapter 4 provides a high resolution signal decomposition for estimating both frequency and amplitude of oscillatory modes in nonstationary multimodal signals. This data-driven approach uses a single signal segment length for estimating the signal parameters for all oscillatory modes

simultaneously. However, empirical evidence suggests the algorithm performs best when using "optimal" segment lengths, where an optimal segment length contains approximately full periods of the oscillatory modes. This presents a challenge for modes that are significantly out of phase, and causes error in the time-frequency analysis of nonstationary signals. Future work may aim to use different segment lengths for different periodic modes to optimize the accuracy of the decomposition algorithm.

BIBLIOGRAPHY

[1] Ivar Stakgold. *Boundary Value Problems of Mathematical Physics: 2-Volume Set*. Vol. 29. Philadelphia: Society for Industrial and Applied Mathematics, 2000.

[2] Ivar Stakgold and Michael J Holst. *Green's Functions and Boundary Value Problems*. Vol. 99. New York: John Wiley & Sons, 2011.

[3] J.D. Hamilton and Princeton University Press. *Time Series Analysis*. Time Series Analysis v. 10. Princeton University Press, 1994.

[4] Daniel E. Shea, Steven L. Brunton, and J. Nathan Kutz. *SINDy-BVP: Sparse Identification of Nonlinear Dynamics for Boundary Value Problems*. May 21, 2020. arXiv: 2005.10756 [cs].

[5] Craig R. Gin et al. *DeepGreen: Deep Learning of Green's Functions for Nonlinear Boundary Value Problems*. Dec. 31, 2020. arXiv: 2101.07206 [physics].

[6] Daniel E. Shea et al. *Extraction of Instantaneous Frequencies and Amplitudes in Nonstationary Time-Series Data*. Apr. 2, 2021. arXiv: 2104.01293 [cs, eess, math].

[7] J.B.J. Fourier and Firmin père fils Didot. *Théorie Analytique de La Chaleur, Par M. Fourier*. chez Firmin Didot, pere et fils, 1822.

[8] Christophe Aristégui and Stéphane Baste. "Optimal Recovery of the Elasticity Tensor of General Anisotropic Materials from Ultrasonic Velocity Data". In: *The Journal of the Acoustical Society of America* 101.2 (Feb. 1997), pp. 813–833.

[9] Baskar Ganapathysubramanian and Nicholas Zabaras. "Modeling Diffusion in Random Heterogeneous Media: Data-Driven Models, Stochastic Collocation and the Variational Multiscale Method". In: *Journal of Computational Physics* 226.1 (Sept. 2007), pp. 326–353.

[10] M. Kachanov, I. Sevostianov, and B. Shafiro. "Explicit Cross-Property Correlations for Porous Materials with Anisotropic Microstructures". In: *Journal of the Mechanics and Physics of Solids* 49.1 (Jan. 2001), pp. 1–25.

[11] Sun K. Kim et al. "Inverse Estimation of Thermophysical Properties for Anisotropic Composite". In: *Experimental Thermal and Fluid Science* 27.6 (July 2003), pp. 697–704.

[12] Ran Bachrach et al. "Reconstruction of the Layer Anisotropic Elastic Parameters and High-Resolution Fracture Characterization from P-Wave Data: A Case Study Using Seismic Inversion and Bayesian Rock Physics Parameter Estimation". In: *Geophysical Prospecting* 57.2 (Mar. 2009), pp. 253–262.

[13] Salvatore Torquato. *Random Heterogeneous Materials*. 1st ed. Vol. 16. Interdisciplinary Applied Mathematics. Springer-Verlag New York, 2002.

[14] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems". In: *Proceedings of the National Academy of Sciences* 113.15 (Apr. 2016), pp. 3932–3937.

[15] T. Kirchdoerfer and M. Ortiz. "Data-Driven Computational Mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 304 (June 2016), pp. 81–101.

[16] T. Kirchdoerfer and M. Ortiz. "Data Driven Computing with Noisy Material Data Sets". In: *Computer Methods in Applied Mechanics and Engineering* 326 (Nov. 2017), pp. 622–641.

[17] Daniel Z. Huang et al. “Learning Constitutive Relations from Indirect Observations Using Deep Neural Networks”. In: *Journal of Computational Physics* 416 (Sept. 2020), p. 109491.

[18] Lu Trong Khiem Nguyen and Marc-André Keip. “A Data-Driven Approach to Nonlinear Elasticity”. In: *Computers & Structures* 194 (Jan. 2018), pp. 97–115.

[19] Adrien Leygue et al. “Data-Based Derivation of Material Response”. In: *Computer Methods in Applied Mechanics and Engineering* 331 (Apr. 2018), pp. 184–196.

[20] M.A. Bessa et al. “A Framework for Data-Driven Analysis of Materials under Uncertainty: Countering the Curse of Dimensionality”. In: *Computer Methods in Applied Mechanics and Engineering* 320 (June 2017), pp. 633–667.

[21] Kailai Xu and Eric Darve. *ADCME: Learning Spatially-Varying Physical Fields Using Deep Neural Networks*. Nov. 24, 2020. arXiv: `2011.11955 [cs, math]`.

[22] Paromita Nath, Zhen Hu, and Sankaran Mahadevan. “Sensor Placement for Calibration of Spatially Varying Model Parameters”. In: *Journal of Computational Physics* 343 (Aug. 2017), pp. 150–169.

[23] P.S. Koutsourelakis. “A Multi-Resolution, Non-Parametric, Bayesian Framework for Identification of Spatially-Varying Model Parameters”. In: *Journal of Computational Physics* 228.17 (Sept. 2009), pp. 6184–6211.

[24] Steven L Brunton and J Nathan Kutz. “Methods for Data-Driven Multiscale Model Discovery for Materials”. In: *Journal of Physics: Materials* 2.4 (2019), p. 044002.

[25] Samuel H. Rudy et al. “Data-Driven Discovery of Partial Differential Equations”. In: *Science Advances* 3.4 (Apr. 2017), e1602614.

[26] Samuel Rudy et al. *Data-Driven Identification of Parametric Partial Differential Equations*. June 2018. arXiv: 1806.00732 [math].

[27] Peng Zheng et al. "A Unified Framework for Sparse Relaxed Regularized Regression: SR3". In: *IEEE Access* 7 (2018), pp. 1404–1423.

[28] Linan Zhang and Hayden Schaeffer. "On the Convergence of the SINDy Algorithm". In: *Multiscale Modeling & Simulation* 17.3 (2019), pp. 948–972.

[29] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. JSTOR: 2346178.

[30] Kathleen Champion et al. *A Unified Sparse Optimization Framework to Learn Parsimonious Physics-Informed Models from Data*. July 2, 2020. arXiv: 1906.10612 [physics].

[31] Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. "Sparse Learning of Stochastic Dynamical Equations". In: *The Journal of chemical physics* 148.24 (2018), p. 241723.

[32] Sheng Zhang and Guang Lin. "Robust Data-Driven Discovery of Governing Physical Laws with Error Bars". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2217 (2018), p. 20180305.

[33] W. Pan et al. "A Sparse Bayesian Approach to the Identification of Nonlinear State-Space Systems". In: *IEEE Transactions on Automatic Control* 61.1 (Jan. 2016), pp. 182–187.

[34] Robert K Niven et al. "Bayesian Identification of Dynamical Systems". In: *Multidisciplinary Digital Publishing Institute Proceedings* 33.1 (2020), p. 33.

[35] Kailiang Wu and Dongbin Xiu. "Numerical Aspects for Approximating Governing Equations Using Data". In: *Journal of Computational Physics* 384 (May 2019), pp. 200–221.

[36] J Nathan Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press, 2013.

[37] Daniel A. Messenger and David M. Bortz. *Weak SINDy: Galerkin-Based Data-Driven Model Selection*. July 3, 2020. arXiv: `2005.04339 [cs, math]`.

[38] Patrick A. K. Reinbold, Daniel R. Gurevich, and Roman O. Grigoriev. "Using Noisy or Incomplete Data to Discover Models of Spatiotemporal Dynamics". In: *Phys. Rev. E* 101.1 (Jan. 2020), 010203(R).

[39] Patrick Gelß et al. "Multidimensional Approximation of Nonlinear Dynamical Systems". In: *Journal of Computational and Nonlinear Dynamics* 14.6 (Apr. 2019).

[40] John David Jackson. *Classical Electrodynamics*. John Wiley & Sons, 2007.

[41] A Yariv. *Quantum Electronics*. John Wiley & Sons, 1989.

[42] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. "Deep Learning for Universal Linear Embeddings of Nonlinear Dynamics". In: *Nature communications* 9.1 (2018), p. 4950.

[43] Kathleen Champion et al. "Data-Driven Discovery of Coordinates and Governing Equations". In: *Proceedings of the National Academy of Sciences* 116.45 (Nov. 2019), pp. 22445–22451.

[44] Christoph Wehmeyer and Frank Noé. "Time-Lagged Autoencoders: Deep Learning of Slow Collective Variables for Molecular Kinetics". In: *The Journal of Chemical Physics* 148.24 (2017), p. 241703.

[45] Andreas Mardt et al. "VAMPnets: Deep Learning of Molecular Kinetics". In: *Nature Communications* 9.1 (Dec. 2018), p. 5. arXiv: `1710.06012`.

[46] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. "Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1130–1140.

[47] Enoch Yeung, Soumya Kundu, and Nathan Hodas. "Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems". In: *2019 American Control Conference (ACC)*. 2019 American Control Conference (ACC). Philadelphia, PA, USA: IEEE, July 2019, pp. 4832–4839.

[48] Samuel E. Otto and Clarence W. Rowley. *Linearly-Recurrent Autoencoder Networks for Learning Dynamics*. Jan. 15, 2019. arXiv: `1712.01378 [cs, math, stat]`.

[49] Qianxiao Li et al. "Extended Dynamic Mode Decomposition with Dictionary Learning: A Data-Driven Adaptive Spectral Decomposition of the Koopman Operator". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.10 (Oct. 2017), p. 103111. arXiv: `1707.00225`.

[50] Carmeline J Dsilva et al. "Parsimonious Representation of Nonlinear Dynamical Systems through Manifold Learning: A Chemotaxis Case Study". In: *Applied and Computational Harmonic Analysis* 44.3 (2018), pp. 759–773.

[51] Craig Gin et al. "Deep Learning Models for Global Coordinate Transformations That Linearize PDEs". In: *European Journal of Applied Mathematics* (Sept. 24, 2020), pp. 1–25. arXiv: `1911.02710`.

[52] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (Dec. 1989), pp. 303–314.

[53] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks". In: *Neural networks* 3.5 (1990), pp. 551–560.

[54] B. O. Koopman. "Hamiltonian Systems and Transformation in Hilbert Space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318.

[55] I. Mezić and A. Banaszuk. "Comparison of Systems with Complex Behavior". In: *Physica D: Nonlinear Phenomena* 197.1–2 (2004), pp. 101–133.

[56] I. Mezić. "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions". In: *Nonlinear Dynamics* 41.1-3 (2005), pp. 309–325.

[57] M. Budišić and I. Mezić. "Geometry of the Ergodic Quotient Reveals Coherent Structures in Flows". In: *Physica D: Nonlinear Phenomena* 241.15 (2012), pp. 1255–1269.

[58] I. Mezić. "Analysis of Fluid Flows via Spectral Properties of the Koopman Operator". In: *Annual Review of Fluid Mechanics* 45 (2013), pp. 357–378.

[59] Steven L. Brunton et al. "Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control". In: *PLOS ONE* 11.2 (2016), pp. 1–19.

[60] Peter J. Schmid. "Dynamic Mode Decomposition of Numerical and Experimental Data". In: *Journal of Fluid Mechanics* 656 (Aug. 10, 2010), pp. 5–28.

[61] C. W. Rowley et al. "Spectral Analysis of Nonlinear Flows". In: *J. Fluid Mech.* 645 (2009), pp. 115–127.

[62] J. N. Kutz et al. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[63] Frank Noé and Feliks Nüske. "A Variational Approach to Modeling Slow Processes in Stochastic Dynamical Systems". In: *Multiscale Modeling & Simulation* 11.2 (2013), pp. 635–655.

[64] Feliks Nüske et al. "Variational Approach to Molecular Kinetics". In: *Journal of chemical theory and computation* 10.4 (2014), pp. 1739–1752.

[65] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. "A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition". In: *Journal of Nonlinear Science* 25 (2015), pp. 1307–1346.

[66] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. "A Kernel-Based Method for Data-Driven Koopman Spectral Analysis". In: *Journal of Computational Dynamics* 2.2 (2015), pp. 247–265.

[67] Stefan Klus et al. "Data-Driven Model Reduction and Transfer Operator Approximation". In: *Journal of Nonlinear Science* 28.3 (2018), pp. 985–1010.

[68] J. N. Kutz, J. L. Proctor, and S. L. Brunton. "Applied Koopman Theory for Partial Differential Equations and Data-Driven Modeling of Spatio-Temporal Systems". In: *Complexity* 2018.6010634 (2018), pp. 1–16.

[69] Jacob Page and Rich R Kerswell. "Koopman Analysis of Burgers Equation". In: *Physical Review Fluids* 3.7 (2018), p. 071901.

[70] C. Bishop. *textslPattern Recognition and Machine Learning*. Springer, 2006.

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[72] R Rico-Martinez, IG Kevrekidis, and K Krischer. "Nonlinear System Identification Using Neural Networks: Dynamics and Instabilities". In: *Neural networks for chemical engineers* (1995), pp. 409–442.

[73] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. "Identification of Distributed Parameter Systems: A Neural Net Based Approach". In: *Computers & Chemical Engineering* 22 (Mar. 1998), S965–S968.

[74] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. "Deep Learning of Dynamics and Signal-Noise Decomposition with Time-Stepping Constraints". In: *Journal of Computational Physics* 396 (2019), pp. 483–506.

[75] Henning Lange, Steven L. Brunton, and Nathan Kutz. *From Fourier to Koopman: Spectral Methods for Long-Term Time Series Prediction*. Apr. 1, 2020. arXiv: 2004.00574 [cs, eess, stat].

[76] Yuying Liu, J. Nathan Kutz, and Steven L. Brunton. *Hierarchical Deep Learning of Multiscale Differential Equation Time-Steppers*. Aug. 22, 2020. arXiv: 2008.09768 [physics].

[77] Shaowu Pan and Karthik Duraisamy. "Physics-Informed Probabilistic Learning of Linear Embeddings of Nonlinear Dynamics With Guaranteed Stability". In: *SIAM Journal on Applied Dynamical Systems* 19.1 (2020), pp. 480–509.

[78] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[79] Stéphane Mallat. "Understanding Deep Convolutional Networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150203.

[80] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[81] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[82] Zhengping Che et al. "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *Scientific Reports* 8.1 (Apr. 17, 2018), p. 6085.

[83] Jaideep Pathak et al. "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach". In: *Physical review letters* 120.2 (2018), p. 024102.

[84] G. Larry Bretthorst. *Bayesian Spectrum Analysis and Parameter Estimation*. Lecture Notes in Statistics 48. New York: Springer-Verlag, 1988. 209 pp.

[85] J. Allen. “Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.3 (June 1977), pp. 235–238.

[86] Guowei Tu et al. “Iterative Nonlinear Chirp Mode Decomposition: A Hilbert-Huang Transform-like Method in Capturing Intra-Wave Modulations of Nonlinear Responses”. In: *Journal of Sound and Vibration* 485 (Oct. 2020), p. 115571.

[87] S. K. Hadjidimitriou and L. J. Hadjileontiadis. “Toward an EEG-Based Recognition of Music Liking Using Time-Frequency Analysis”. In: *IEEE Transactions on Biomedical Engineering* 59.12 (2012), pp. 3498–3510.

[88] Marián Képesi and Luis Weruaga. “Adaptive Chirp-Based Time–Frequency Analysis of Speech Signals”. In: *Speech Communication* 48.5 (2006), pp. 474–492.

[89] Leicheng Guo et al. “River-Tide Dynamics: Exploration of Nonstationary and Nonlinear Tidal Behavior in the Yangtze River Estuary: River Tidal Dynamics”. In: *Journal of Geophysical Research: Oceans* 120.5 (May 2015), pp. 3499–3521.

[90] Y. Lin and M. Tsai. “Development of an Improved Time–Frequency Analysis-Based Nonintrusive Load Monitor for Load Demand Identification”. In: *IEEE Transactions on Instrumentation and Measurement* 63.6 (2014), pp. 1470–1483.

[91] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. 3rd ed. Cambridge University Press, 2018.

[92] Norden E. Huang et al. “The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series Analysis”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1971 (Mar. 8, 1998), pp. 903–995.

[93] Dennis Gabor. “Theory of Communication. Part 1: The Analysis of Information”. In: *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering* 93.26 (1946), pp. 429–441.

[94] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

[95] Konstantin Dragomiretskiy and Dominique Zosso. “Variational Mode Decomposition”. In: *IEEE Transactions on Signal Processing* 62.3 (Feb. 2014), pp. 531–544.

[96] Shiqian Chen et al. “Nonlinear Chirp Mode Decomposition: A Variational Method”. In: *IEEE Transactions on Signal Processing* 65.22 (Nov. 2017), pp. 6024–6037.

[97] Pushpendra Singh et al. “The Fourier Decomposition Method for Nonlinear and Non-Stationary Time Series Analysis”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2199 (Mar. 2017), p. 20160871.

[98] Matthieu Kowalski, Adrien Meynard, and Hau-tieng Wu. “Convex Optimization Approach to Signals with Fast Varying Instantaneous Frequency”. In: *Applied and Computational Harmonic Analysis* 44.1 (Jan. 2018), pp. 89–122.

[99] Seth M Hirsh, Bingni W Brunton, and J Nathan Kutz. “Data-Driven Spatiotemporal Modal Decomposition for Time Frequency Analysis”. In: *Applied and Computational Harmonic Analysis* 49.3 (2020), pp. 771–790.

[100] Thomas Y Hou and Zuoqiang Shi. “Sparse Time-Frequency Decomposition for Multiple Signals with Same Frequencies”. In: *Advances in Data Science and Adaptive Analysis* 9.04 (2017), p. 1750010.

[101] Rajiv Giridharagopal et al. “Submicrosecond Time Resolution Atomic Force Microscopy for Probing Nanoscale Dynamics”. In: *Nano Letters* 12.2 (Feb. 8, 2012), pp. 893–898.

[102] Mehrdad Yazdanian, Ali Mehrizi-Sani, and Mohsen Mojiri. "Estimation of Electromechanical Oscillation Parameters Using an Extended Kalman Filter". In: *IEEE Transactions on Power Systems* 30.6 (Nov. 2015), pp. 2994–3002.

[103] Ryan P. Dwyer, Lee E. Harrell, and John A. Marohn. "Lagrangian and Impedance-Spectroscopy Treatments of Electric Force Microscopy". In: *Physical Review Applied* 11.6 (June 10, 2019), p. 064020.

[104] Ingrid Daubechies, Jianfeng Lu, and Hau-Tieng Wu. *Synchrosqueezed Wavelet Transforms: A Tool for Empirical Mode Decomposition*. Dec. 12, 2009. arXiv: `0912.2437 [math]`.

[105] Z.K. Peng, Peter W. Tse, and F.L. Chu. "An Improved Hilbert–Huang Transform and Its Application in Vibration Signal Analysis". In: *Journal of Sound and Vibration* 286.1-2 (Aug. 2005), pp. 187–205.

[106] Yangbo Chen and Maria Q. Feng. "A Technique to Improve the Empirical Mode Decomposition in the Hilbert-Huang Transform". In: *Earthquake Engineering and Engineering Vibration* 2.1 (June 2003), pp. 75–85.

[107] Naveed ur Rehman and Hania Aftab. "Multivariate Variational Mode Decomposition". In: *IEEE Transactions on Signal Processing* 67.23 (Dec. 1, 2019), pp. 6039–6052.

[108] Michael Feldman. "Non-Linear System Vibration Analysis Using Hilbert Transform–I. Free Vibration Analysis Method 'Freevib'". In: *Mechanical Systems and Signal Processing* 8.3 (Mar. 1994), pp. 119–127.

[109] Michael Feldman. "Non-Linear System Vibration Analysis Using Hilbert Transform–II. Forced Vibration Analysis Method 'Forcevib'". In: *Mechanical Systems and Signal Processing* 8.3 (May 1994), pp. 309–318.

[110] Michael Feldman. "Hilbert Transform in Vibration Analysis". In: *Mechanical Systems and Signal Processing* 25.3 (Apr. 2011), pp. 735–802.

[111] Ljubiša Stankovic et al. “On the Decomposition of Multichannel Nonstationary Multicomponent Signals”. In: *Signal Processing* (2020), p. 13.

[112] R. Yan and R. X. Gao. “Hilbert–Huang Transform-Based Vibration Signal Analysis for Machine Health Monitoring”. In: *IEEE Transactions on Instrumentation and Measurement* 55.6 (Dec. 2006), pp. 2320–2329.

[113] C. D. Saragiotis, L. J. Hadjileontiadis, and S. M. Panas. “A Higher-Order Statistics-Based Phase Identification of Three-Component Seismograms in a Redundant Wavelet Transform Domain”. In: *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics. SPW-HOS ’99*. Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics. SPW-HOS ’99. June 1999, pp. 396–399.

[114] Rajiv Giridharagopal et al. “Time-Resolved Electrical Scanning Probe Microscopy of Layered Perovskites Reveals Spatial Variations in Photoinduced Ionic and Electronic Carrier Motion”. In: *ACS Nano* 13.3 (Mar. 26, 2019), pp. 2812–2821.

[115] Ilias Pagkalos et al. “A High-Performance Application Specific Integrated Circuit for Electrical and Neurochemical Traumatic Brain Injury Monitoring”. In: *Chemphyschem* 19.10 (May 22, 2018), pp. 1215–1225. pmid: 29388305.

[116] Gaozhi Xiao and Wojtek J. Bock. *Photonic Sensing: Principles and Applications for Safety and Security Monitoring*. 1st ed. Wiley Series in Microwave and Optical Engineering. NJ: John Wiley & Sons, Inc., 2012.

[117] Weilin Liu et al. “A Fully Reconfigurable Photonic Integrated Signal Processor”. In: *Nature Photonics* 10.3 (Mar. 2016), pp. 190–195.

[118] David C. Coffey and David S. Ginger. “Time-Resolved Electrostatic Force Microscopy of Polymer Solar Cells”. In: *Nature Materials* 5.9 (9 Sept. 2006), pp. 735–740.

[119] Ryan P. Dwyer, Sarah R. Nathan, and John A. Marohn. "Microsecond Photocapacitance Transients Observed Using a Charged Microcantilever as a Gated Mechanical Integrator". In: *Science Advances* 3.6 (June 2017), e1602951.

[120] Aaron Mascaro et al. "Review of Time-Resolved Non-Contact Electrostatic Force Microscopy Techniques with Applications to Ionic Transport Measurements". In: *Beilstein Journal of Nanotechnology* 10.1 (Mar. 1, 2019), pp. 617–633.

[121] S.V. Kalinin and A. Gruverman. *Scanning Probe Microscopy: Electrical and Electromechanical Phenomena at the Nanoscale*. Scanning Probe Microscopy: Electrical and Electromechanical Phenomena at the Nanoscale v. 1. Springer New York, 2007.

[122] Jeffrey S. Harrison et al. "Noncontact Imaging of Ion Dynamics in Polymer Electrolytes with Time-Resolved Electrostatic Force Microscopy". In: *ACS Nano* 13.1 (Jan. 22, 2019), pp. 536–543.

[123] Rajiv Giridharagopal, Phillip A. Cox, and David S. Ginger. "Functional Scanning Probe Imaging of Nanostructured Solar Energy Materials". In: *Accounts of Chemical Research* 49.9 (Sept. 20, 2016), pp. 1769–1776.

[124] Durmus U. Karatay et al. "Fast Time-Resolved Electrostatic Force Microscopy: Achieving Sub-Cycle Time Resolution". In: *Review of Scientific Instruments* 87.5 (May 2016), p. 053702.

[125] Ali Moeed Tirmzi et al. "Light-Dependent Impedance Spectra and Transient Photoconductivity in a Ruddlesden–Popper 2D Lead–Halide Perovskite Revealed by Electrical Scanned Probe Microscopy and Accompanying Theory". In: *The Journal of Physical Chemistry C* 124.25 (June 25, 2020), pp. 13639–13648.

[126] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. “Sparse Identification of Nonlinear Dynamics with Control (SINDYc)”. In: *IFAC-PapersOnLine* 49.18 (2016), pp. 710–715.