Computational methods for system identification and data-driven forecasting

Samuel Rudy

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

Doctor of Philosophy

University of Washington

2019

Reading Committee:

J. Nathan Kutz, Chair

Steven Brunton, Chair

Eric Shea-Brown

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

Computational methods for system identification
and data-driven forecasting

Samuel Rudy

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

Associate Professor Steven Brunton
Mechanical Engineering

This thesis develops several novel computational tools for system identification and data-driven forecasting. The material is divided into four chapters: data-driven identification of partial differential equations, neural network interpolation of velocity field data from trajectory measurements, smoothing of high dimensional nonlinear time series, and an application of data-driven forecasting in biology.

We first develop a novel computational method for identifying partial differential equations (PDEs) from measurements in the spatio-temporal domain. Building on past methods in sparse regression, we formulate a regression problem to select the active terms of a PDE from a large library of candidate basis functions. In contrast to many data-driven forecasting methods, the proposed algorithm yields exact representations of the dynamics. This has the advantage of allowing for future state prediction from novel initial and boundary conditions as well as rigorous mathematical analysis. The method is also extended to the case where coefficients vary either in space or time. We demonstrate the ability to accurately learn the correct active terms and their magnitudes on a variety on canonical partial differential equations.

We also develop a method for interpolating the velocity fields of smooth dynamical systems using neural networks. We specifically focus on addressing the issue of learning from noisy and limited data. We construct a cost function for training neural network interpolations of velocity fields from trajectory measurements that explicitly accounts for measurement noise. The need to numerically differentiate data is avoided by placing the neural network interpolation of velocity within an explicit timestepping scheme and training as a flow map rather than directly on the velocity field. The proposed framework is shown to be capable of learning accurate forecasting models even when data is corrupted by significant levels of noise. We also consider some limitations of using neural networks as forecasting models for dynamical systems. Using test problems with known dynamics, we show that neural networks are able to accurately interpolate a vector field only where data is collected and generally exhibit high generalization error. Some guidelines are proposed regarding the contexts in which neural networks may or may not be useful in practice.

For datasets where dynamics are known either completely or up to a set of parameters, we develop a novel smoothing technique based on soft-adherence to governing equations. The proposed method may be applicable to smoothing data from deterministic dynamical systems where high dimensionality or nonlinearity make sequential Bayesian methods impractical. We test the method on several canonical problems from data assimilation and show that it is robust to exceptionally high levels of noise as well as noise with non-zero mean and temporally autocorrelated noise.

The last section of this thesis develops a data-driven forecasting model for the half-sarcomere, a small component of skeletal muscle tissue. Current models of the half-sarcomere currently require computationally expensive Monte Carlo simulations to resolve the effects of filament compliance. We seek to replicate the dynamic behavior realized by Monte Carlo simulation of the half-sarcomere at a lower cost. Drawing inspiration

from surrogate and reduced order modeling, we apply a course graining to the variables tracked by the Monte Carlo simulation and learn a dynamics model on the course grained variables using data. We find that the resulting data-driven model effectively reproduces force traces and dynamics of the course grained state when given novel input parameters.

Taken together, the innovations presented in this thesis represent a modest contribution to the field of data-driven methods for system identification and forecasting. In the concluding chapter, we highlight several exciting directions that build upon and improve the research presented in this thesis.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables x

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Previous work in data-driven methods 2
1.3 Organization and Contributions 5

Chapter 2: Sparse identification of partial differential equations 9
2.1 Background 9
2.2 Constant Coefficient Equations 10
2.3 Parametric Equations 38
2.4 Discussion 49

Chapter 3: Robust neural network based forecasting 51
3.1 Introduction 51
3.2 Methods 52
3.3 Results 64
3.4 Cautionary remarks on neural networks and overfitting 80
3.5 Discussion 82

Chapter 4: Smoothing by soft adherence to governing equations 84
4.1 Introduction 84
4.2 Methods 85
4.3 Results 89
4.4 Discussion 95

Chapter 5: Data-driven surrogate model of muscle dynamics 98
5.1 Introduction 98
5.2 Methods 99
5.3 Results 108
5.4 Discussion 110

Chapter 6: Conclusions 113
6.1 Future Directions 115

Bibliography 117

LIST OF FIGURES

Figure Number Page

2.1 Steps in the PDE functional identification of nonlinear dynamics (PDE-FIND) algorithm, applied to infer the Navier-Stokes equation from data. **1a.** Data is collected as snapshots of a solution to a PDE. **1b.** Numerical derivatives are taken and data is compiled into a large matrix Θ, incorporating candidate terms for the PDE. **1c.** Sparse regressions is used to identify active terms in the PDE. **2a.** For large datasets, sparse sampling may be used to reduce the size of the problem. **2b.** Subsampling the dataset is equivalent to taking a subset of rows from the linear system in (2.3). **2c.** An identical sparse regression problem is formed but with fewer rows. **d.** Active terms in ξ are synthesized into a PDE. 17

2.2 The numerical solution to the KdV equation. 21

2.3 The numerical solution to the Burgers' equation. 23

2.4 The magnitude of the numerical solution to the Schrödinger's equation. . . 24

2.5 The magnitude of the numerical solution to the nonlinear Schrödinger's equation. 25

2.6 The numerical solution to the Kuramoto-Sivashinsky equation. 27

2.7 The numerical solution to the reaction diffusion equation. 29

2.8 A single snapshot of the vorticity field is illustrated for the fluid flow past a cylinder. The sampling region is outlined in red. 30

2.9 Inferring the diffusion equation from a single Brownian motion. (**a**) Time series is broken into many short random walks that are used to construct histograms of the displacement. (**b**) A single stochastic realization of Brownian motion. (**c**) Parameter error ($\|\xi^* - \hat{\xi}\|_1$) vs. length of known time series. Blue symbols correspond to correct identification of the structure of the diffusion model, $u_t = cu_{xx}$. 31

2.10 Five empirical distributions are presented illustrating the statistical spread of a particle's expected location over time. The data used to construct these histograms is collected from a single Brownian motion trajectory, see Figure 2.9b. 32

2.11 Five empirical distributions are presented illustrating the statistical spread of a particle's expected location over time with constant drift. 33
2.12 The numerical solution to the misidentified Kuramoto-Sivashinsky equation given by (2.14). 34
2.13 The numerical solution to the misidentified Nonlinear Schrödinger equation given by (2.15). 35
2.14 Results of PDE-FIND applied to Burgers' equation for varying levels of noise. 36
2.15 Two prototypical scenarios for parametric model discovery: (a) Parameters $\mu(t)$ are piece wise constant and change at fixed points in time, (b) The underlying PDE model $u_t = N(u, \mu(t))$ depends on continuously varying parameter $\mu(t)$. 38
2.16 Example of loss function evaluated for a number of candidate models for the parametric Burgers' equation. Library included derivatives up to 4th order multiplying powers of u up to cubic. Left: 50 models obtained via SGTR algorithm using values of ϵ between ϵ_{min} and ϵ_{max}. Right: 50 models obtained via group LASSO for λ between $10^{-5}\lambda_{max}$ and λ_{max}. 43
2.17 Left: dataset for identification of the parametric diffusive Burgers' equation. Here the PDE was evolved on the interval $[-8, 8]$ with periodic boundary conditions and $t \in [0, 10]$. Right: Coefficients for the terms in the parametric Burgers' equation. The diffusion was held constant at 0.1 while the nonlinear advection as coefficient is given by $a(t) = -(1 + sin(t)/4)$. 44
2.18 Time series discovered for the coefficients of the parametric Burgers' equation. Top row: SGTR method, which correctly identifies the two terms. Bottom row: group LASSO method which adds several additional (incorrect) terms to the model. The left panels are noise-free, while the right panels contain 1% noise. 45
2.19 Left: dataset for identification of the parametric Navier-Stokes equation (2.26). Right: coefficients for Navier-Stokes equations exhibiting jump in Reynolds number from 100 to 75 at $t = 10$. This parametric dependency is illustrated in Fig. 2.15a. 47
2.20 Identified time series for coefficients for the Navier-Stokes equation. Distinct axes are used to highlight jump in Reynolds number. Left: no noise. Right: 1% noise 47
2.21 Left: dataset for identification of the spatially dependent advection diffusion equation. Right: Spatial dependence of PDE. In this case, the loadings $\xi_j(t)$ in (2.18) are replaced by $\xi_j(x)$. 48

2.22 Spatial dependence of advection diffusion equation. Left: no noise. Right: 1% noise. Both SGTR and group LASSO correctly identified the active terms. 48

2.23 Left: dataset for identification of the spatially dependent Kuramoto-Sivashinsky equation. Right: parametric dependency of the governing equations. 49

2.24 Spatial dependence of Kuramoto-Sivashinsky. top row: SGTR. Bottom row: group LASSO. Left: no noise. Right 0.01% noise using SGTR. SGTR detects correct sparsity with significant parameter error. Group LASSO does not correctly identify the parsimonious model, nor does it do a good job at predicting the correct parametric values. 50

3.1 Measurements of a dynamical system may be split into measurement error and an underlying state that is governed by a dynamical system. 53

3.2 Schematic of de-noising framework for constrained learning of nonlinear system identification. **(a)** Flow of full dataset through (3.4) to find forward and backward measurements. Measurements are decomposed into learned measurement error and state, the state is propagated forward and backward using learned dynamics, and measurement error is added back to find forward and backward measurements and evaluate model accuracy. **(b)** Example of data-driven estimate of forward propagation of single observation from graph in panel a. **(c)** Example 3-step Runge-Kutta scheme to approximate the map F and isolate the vector field, which is modeled by a feed-forward neural network. 55

3.3 Results for stable spiral with several network architectures and noise magnitudes. Top row: Learned vector field $\hat{f}_{\boldsymbol{\theta}}$ for each configuration. Middle row: ℓ^2 error of learned vector field plotted with training data $\mathbf{Y}$ in blue and true state $\mathbf{X}$ in black. Bottom row: True x-component of measurement error $\boldsymbol{\nu}$ in blue and learned $\hat{\boldsymbol{\nu}}$ in red, with difference as dashed black line. . . 65

3.4 Results for cubic oscillator with increasing magnitudes of measurement noise. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of x_1 according to data-driven dynamics $\hat{F}_{\boldsymbol{\theta}}$ in green. Right column: Learned neural network representation of vector field $\hat{f}_{\boldsymbol{\theta}}$. True vector field is shown in top right for comparison. 67

3.5 Results from a single trial of the cubic oscillator with 10% noise. Left: True measurement noise (blue), learned measurement noise (red), and error (dotted black line). Right: Heat map of vector field approximation error with the noiseless training trajectory plotted for reference. Note the significant increase in error magnitude in regions far from the training data. . . . 68

3.6 Average error over 50 trials and standard deviations for the cubic oscillator at varying levels of measurement noise. Left: Mean E_N and shaded region indicating one standard deviation. Center: Mean E_f and standard error. Right: Median normalized ℓ^2 difference between true trajectory and forward orbit under $\hat{f}_{\boldsymbol{\theta}}$, given by E_F. The distribution for 18% noise is omitted due to a single unstable trajectory resulting in non-numeric data; however, the median is reported. 69

3.7 Results for Lorenz system with increasing magnitudes of measurement noise. Top row: Clean data. Second row: 1% Gaussian noise. Bottom row: 10% noise from Student's T distribution with 10 degrees of freedom. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. The prediction is extended to T_{max} five times that of training data. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in noise estimate in dashed black. 70

3.8 Histograms showing true and learned sample distribution of measurement noise with distribution of measurement noise plotted in black. Left: Learned noise from Lorenz system with 10% Gaussian distributed noise. Right: Learned noise from Lorenz system with 10% noise from Student's T distribution with 10 degrees of freedom. 71

3.9 Marginal phase space densities and autocorrelation function for Lorenz system. 72

3.10 Modes from simulation of Navier-Stokes equation. The x, y, z coordinates used in this test problem are time series obtained by projecting the full flow field onto POD mode 1, POD mode 2, and the shift mode, respectively. . . . 73

3.11 Results for reduced basis of flow around a cylinder with increasing magnitudes of measurement noise. Top row: Clean data. Bottom row: 1% Gaussian noise. Left column: Observations Y in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. Note green trajectory has been extended to T_{max} five times that of training data. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in estimation of measurement noise as dashed black line. 74

3.12 Results for double pendulum with increasing magnitudes of measurement noise. Top row: Clean data. Second row: 5% Gaussian noise. Bottom row: 10% Gaussian noise. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in estimation of measurement noise as dashed black line. 76

3.13 Error metrics and training time for varying q. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36). 77

3.14 Error metrics and training time for varying regularization γ on measurement noise estimates. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36). 78

3.15 Error metrics and training time for varying regularization β on neural network weights. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36). 79

3.16 Error metrics and training time for varying geometric decay constant ρ^{-1} for multistep prediction loss. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36). 79

3.17 Learned vector fields for the Lorenz system with 5% noise. Left: Single trajectory (top) and learned vector field (bottom) at Poincare section $z = 25$. xy components of $\hat{f}_{\boldsymbol{\theta}}$ shown in stream plot with z component given by color. Center: 50 short trajectories and learned vector field using (3.40). Right: True vector field. 81

3.18 Learned vector fields for the reduced order system from flow around a cylinder with different sizes of neural network. Left: 3 rows of 64 nodes each. Center: 3 rows of 128 nodes. Right: 5 rows of 64 nodes. Mean field model exhibits radial symmetry. 82

4.1 Denoising results for Lorenz-63 system corrupted by white noise with non-zero mean. $\nu \sim \mathcal{N}(\mu, \Sigma_X^2)$ with $\mu = (5, -5, -5)$. RMSE = 0.397 90

4.2 Denoising results for Lorenz-63 system with $\sigma_\nu^2 = \sigma_X^2$ according to various distributions. Blue: Runge-Kutta denoising with known dynamics, Red: Runge-Kutta denoising with unknown (μ, σ, ρ), Green: Ensemble RTS Smoother with $N_e = 500$. 90

4.3 Denoising and parameter estimation results for Lorenz-96 system with 100% white noise. $\hat{F} = 15.94$ 91

4.4 Comparison between RKD and EnRTS for Lorenz-96 system with 100% white noise. Top left: X. Top right: Y. Center left: X_{RKD}. Center right: X_{EnRTS}. Bottom left: $|X - X_{RKD}|$. Bottom right: $|X - X_{EnRTS}|$. 92
4.5 Example of data and smoothed time series using RKD and EnRTS for Lorenz-96 system with 100% white noise. 93
4.6 Error for denoising Lorenz-96 system with 100% white noise using EnRTS, RKD with known parameters, and RKD with parameter estimation. 94
4.7 Results on Kuramoto-Sivashinsky equation. top left: X. Top right: Y with $\sigma_N^2 = 5\sigma_X^2$. Bottom left: $\hat{X}_{RKD}$. Bottom right: $|X - \hat{X}_{RKD}|$. 95
4.8 Example of data and smoothed time series using RKD for KS system with 500% red noise ($\rho = 0.75$). 96
4.9 Root mean square error for the solution of (4.7) across 25 trials for various values of α in red. RMSE for (4.11) shown in blue. 96
4.10 Root mean square error for the solution of (4.11) using $g(\cdot) = \lambda\|\cdot\|_1$ or $g(\cdot) = \lambda\|\cdot\|_2^2$ for various λ averaged over 10 random noise samples on the Lorenz 63 and Lorenz 96 systems. λ on the order of $10^{-8} - 10^{-4}$ provides small advantage over $\lambda = 0$ solution. 97

5.1 Schematic figure for data-driven modeling of muscle. (left) data is collected from MC simulation of full system. (middle) transition probabilities are modeled as a Markov matrix having edge weights dependent on input and radial tension. Forcing is modeled as function of input and number of cross bridges in each state. (right) A new instance of input parameterization is considered, Markov transition probabilities give states of cross bridges, and force and tension are then computed. 103
5.2 Example of empirical transition probabilities from mechanistic model in blue and computed transition probabilities from data-driven model in red for input regime in testing data. 108
5.3 Example of crossbridge state averages for the mechanistic model in blue and data-driven model in red for parameter regime in testing data. Mechanistic data is averaged over 20 runs with shaded region indicating one standard deviation. 109
5.4 Example of forcing for the mechanistic model in blue and data-driven model in red for parameter regime in testing data. Mechanistic data is averaged over 20 runs with shaded region indicating one standard deviation. 110
5.5 Computed workloops for testing data according to mechanistic model in blue and data-driven model in red. Each plot corresponds to an input regime not seen during the training procedure. 111

5.6 Computed work in mechanistic model on x-axis and data-driven model on y-axis. Blue data points correspond to input in training data while red is testing. Error bars indicate one standard deviation around mean value observed over 20 trials of the mechanistic model. 112

LIST OF TABLES

Table Number Page

2.1 Summary of regression results for a wide range of canonical models of mathematical physics. In each example, the correct model structure is identified using PDE-FIND. The spatial and temporal sampling used for the regression is given along with the error produced in the parameters of the model for both no noise and 1% noise. In the reaction-diffusion (RD) system, 0.5% noise is used. For Navier-Stokes and Reaction Diffusion, the percent of data used in subsampling is also given. 20

2.2 Summary of PDE-FIND for identifying the KdV equation. 21

2.3 Summary of PDE-FIND for identifying Burgers' equation. 23

2.4 Summary of PDE-FIND for identifying the Schrödinger equation. 24

2.5 Summary of PDE-FIND for identifying the Nonlinear Schrödinger equation. 25

2.6 Summary of PDE-FIND for identifying the Kuramoto Sivashinksy equation. 26

2.7 Summary of PDE-FIND for identifying reaction diffusion equation. 28

2.8 Summary of PDE-FIND for identifying the Navier-Stokes equation. 29

2.9 Accuracy of PDE-FIND on Burger's equation with various grid sizes. Red table entries denote a misidentification of the sparisty pattern either due to the inclusion of extra terms or missing one of the two terms in Burgers' equation. Blue entries show average parameter error as percent of true value. Measurements on all grids were taken from numerical solution on fine grid to ensure error in the method is intrinsic to PDE-FIND and not the numerical solution of Burgers' equation. 37

3.1 Error for linear test case with varying levels of noise and neural network structures. 66

3.2 Error for cubic oscillator model with varying noise. 68

3.3 Error for Lorenz system with varying noise. 71

3.4 Moments of analytic, empirical, and learned measurement noise in x-coordinate. 72

3.5 Error for double pendulum with varying noise. 77

ACKNOWLEDGMENTS

I am grateful for the support and guidance that my advisers Nathan Kutz and Steve Brunton have provided over the past 5 years. I have learned a tremendous amount from them regarding the material presented in this dissertation, as well as writing, academic life, and managing stress. Nathan and Steve have always been patient and supportive, and their mentorship has been the highlight of my time at the University of Washington.

In addition to Nathan and Steve, I owe my thanks to several other mentors and collaborators. I am thankful to have had the opportunity to work with Josh Proctor on sparse regression methods. Josh has continued to be someone I can talk to and look up to through out my time at UW. Tom Daniel was a supportive and extremely patient supervisor of the research in this dissertation on the muscle. I also benefited from the guidance of Dave Williams on the same project. The work in this dissertation on discovery of parametric equations was done in collaboration with Alessandro Alla. I would also like to thank Lauren Lederer, who has been a great guide through the process of getting a Ph.D. for myself and many other students.

It has been a pleasure working with other graduate students and post-docs in Nathan and Steve's groups. In particular, I have had insightful conversations and learned a considerable amount from Charlie Fiesler, Kathleen Champion, Floris van Breugel, Krithika Manohar, Niall Mangan, Pedro Maia, and Noah Brenowitz. My family and friends have listened to and been supportive through endless talk of graduate school stress and confusion. I would like to especially thank my parents, brother, and Jentien for being supportive throughout the process of working on my Ph.D.

Chapter 1
INTRODUCTION

Data-driven methods seek to learn physical laws, predictive algorithms, and approximate unknown quantities from measurements using naive assumptions regarding the desired learning objective. These methods are ubiquitous throughout the history of computational science and include the Kalman filter, system identification algorithms, domain decomposition techniques, and much of modern machine learning. This dissertation seeks to develop and analyze several novel data-driven methods in the context of physical problems. We discuss methods for learning exact forms of partial differential equations (PDEs) from data, approximating dynamical systems with neural networks, smoothing noisy data with partially known governing equations, and developing fast surrogate models where the true governing dynamics are computationally expensive.

1.1 Motivation

Dynamical systems are ubiquitous across nearly all fields of science and engineering. When the governing equations of a dynamical system are known, they allow for forecasting, estimation, control, and the analysis of structural stability and bifurcations. Accordingly, substantial scientific effort is directed towards discovering dynamical systems models for physical phenomena. These models have historically been derived via first principles, such as conservation laws or the principle of least action, but these derivations may be intractable for complex systems, in cases where mechanisms are not well understood, and/or when measurements are corrupted by noise. These complex cases motivate automated methods to develop dynamical systems models from data. This is known as system identification.

Conversely one may have access to noisy and/or incomplete measurement of a dynamical system with known or partially known dynamics. Examples include state measurements perturbed by sensor error, or partial measurements of high dimensional systems such as weather. For simple cases, a naive smoother may remove noise from a time series measurement and missing data may be inferred or even ignored, but these techniques often fall short of the requirements for practical implementation. More sophisticated methods use known dynamics of the system being measured and prior assumptions of the measurement and process noise to infer state estimates by balancing what is known from measurements with what is known from the dynamics model. This is known as data assimilation.

Data is also commonly used to simplify complex models or computer codes. In science and engineering, governing equations derived from physical laws often yield intractably complex computational problems. Canonical examples are turbulent fluid flows and simulations constructed from many interacting agents. If high fidelity measurements or simulation data are available and governing equations are either unknown or prohibitively complex then data is often used to construct simpler surrogate models for complex phenomena. This is known as reduced order or surrogate modeling.

1.2 Previous work in data-driven methods

In this section we summarize some of the key contributions in data driven methods that have made the work in this dissertation possible.

1.2.1 System Identification

Data-driven system identification has a rich history in science and engineering [71, 89, 11], and recent advances in computing power and the increasing rate of data collection have led to renewed and expanded interest. Early methods identified linear models from input–output data based on the minimal realization theory of Ho and Kalman [59], in-

cluding the eigensystem realization algorithm (ERA) [72, 94], and the observer/Kalman filter identification (OKID) [74, 124, 123], which are related to the more recent dynamic mode decomposition (DMD) [149, 165, 85]. OKID explicitly accounts for measurement noise by simultaneously identifying a de-noising Kalman filter [77] and the impulse response of the underlying noiseless system, providing considerable noise robustness for linear systems from limited measurements.

Several approaches have also been considered for learning interpretable nonlinear models. Sparse regression techniques have been used to identify exact expressions for nonlinear ordinary differential equations [171, 18, 90, 145, 148, 161, 104], partial differential equations [144, 139, 138], and stochastic differential equations [13]. Much of this work, including the contents of this dissertation, was motivated by the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm, developed in [18]. Recent work has also used Gaussian process regression to obtain exact governing equations [130]. These methods provide interpretable forms for the governing equations but rely on libraries of candidate functions and therefore have difficulty expressing complex dynamics. Symbolic regression [12, 151] allows for more expressive functional forms at the expense of increased computational cost; these methods have been used extensively for automated inference of systems with complex dynamics [152, 39, 40].

A critical challenge in any system identification algorithm is robustness to noisy data. Several of the aforementioned techniques have specific considerations for measurement noise. In particular, the sparse regression methods in [18, 139] use smoothing methods in their numerical differentiation schemes, [161] identifies highly corrupted measurements, [145] attenuates error using integral terms, and [130] naturally treats measurement noise by representing data as a Gaussian process, allowing uncertainty via non-zero covariance in the representation of measurements.

Although the interpretable nonlinear models above are appealing, there are currently limitations to the flexibility of functional forms and the number of degrees of freedom that can be modeled. There has been considerable and growing recent interest in lever-

aging powerful black-box machine learning techniques, such as deep learning, to model increasingly complex systems. More recently, neural networks have been analyzed as dynamical systems [173, 27]. Deep learning is particularly appealing because of the ability to represent arbitrarily complex functions, given enough training data of sufficient variety and quality. Neural networks have been used to model dynamical systems for decades [24, 53, 111], although recent advances in computing power, data volumes, and deep learning architectures have dramatically improved their capabilities. Recurrent neural networks naturally model sequential processes and have been used for forecasting [167, 118, 97, 121, 120, 126] and closure models for reduced order models [170, 117]. Deep learning approaches have also been used recently to find coordinate transformations that make strongly nonlinear systems approximately linear, related to Koopman operator theory [157, 179, 172, 105, 98]. Feed-forward networks may also be used in conjunction with classical methods in numerical analysis to obtain discrete timesteppers [140, 134, 132, 133, 129, 53]. Many of these modern identification and forecasting methods may be broadly cast as nonlinear autoregressive moving average models with exogenous inputs (NARMAX) [25, 11] with increasingly sophisticated interpolation models for the dynamics.

1.2.2 Data Assimilation

The modern analysis of high-dimensional dynamical systems via data assimilation [44] typically leverages a combination of simulation and experimental data, which is often obfuscated by noisy measurement [86, 87]. Noise is well known to critically undermine one's ability to produce accurate low-dimensional diagnostic features, such as POD (proper orthogonal decomposition) [60] or DMD (dynamic mode decomposition) [85] modes, produce accurate forecasts and parameter estimations, generate reduced order models [7], or compute balanced truncation models for control [113, 174, 137]. The ubiquity of data imbued with noise has led to significant research efforts for filtering and parameter estima-

tion in high dimensional chaotic dynamical systems, especially in application areas such as climate modeling, material science, fluid dynamics, and atmospheric sciences. Past method for filtering include sequential Bayesian methods like the Kalman and particle filters [30] and variational methods such as 3D-var and 4D-var [34]. Parameter estimation for noisy data may be carried out by EM (expectation maximization) algorithms [112] alternating between state estimation and parameter estimation. Central to each of these methods is balancing fidelity of state estimates to the prescribed dynamics and to measurements.

Sequential Bayesian filters have gained immense popularity since their introduction by Kalman in 1960 [77]. Kalman filters obtain state estimates by tracking second order statistics of the error covariance for a trajectory with uncertain initial conditions, disturbances, and measurement error. The Extended Kalman Filter (EKF) introduced methodology to filter signals from nonlinear dynamical systems [107] via linearizing the dynamics at each timestep to approximate changes in the error covariance. However, the linearization used in the EKF may fail due to the first order expansion being inaccurate and may also be computationally intractable for large systems. An improved Kalman filter for nonlinear dynamics called the Unscented Kalman Filter (UKF) was introduced in [75] using the unscented transform to track error covariance by using the single timestep trajectories of a set of points chosen to mimic the initial state's error statistics. The Ensemble Kalman Filter (EnKF) [43] was introduced for high dimensional dynamical systems and approximates statistics of the state estimate via a small ensemble of trajectories, generally numbering fewer than the dimension of the state space. Each iteration of the Kalman filter uses Gaussian statistics to track the dynamics. The particle filter was introduced for filtering problems on highly nonlinear systems where Gaussian statistics fail to accurately represent the error [100]. Unfortunately, the particle filter exhibits poor performance in the high dimensional setting [6]. Recent work has developed particle filters for higher dimensional systems [101, 28], but full consideration of non-Gaussian statistics is only considered in a low dimensional space.

State estimates in Kalman filters are conditioned on previous measurements and initial condition. For fixed interval smoothing problems where full trajectory knowledge is available and state estimates do not need to be made in an online manner, the Kalman smoother [45] and the Rauch-Tung-Striebel (RTS) smoother introduce a backwards pass to the Kalman filter to condition state estimates on full trajectory measurements. The RTS smoother is easily extended to nonlinear systems via the Unscented Rauch-Tung-Striebel filter (URTS) [142] and Ensemble Rauch-Tung-Striebel filter (EnRTS) [128]. Accuracy of the RTS smoother is contingent on the forward pass of the underlying Kalman filter. Thus, smoothing of high dimensional nonlinear dynamical systems remains challenging.

1.2.3 Biophysical models of muscle

Biophysical models of muscle contraction are based in sliding filament theory, which was introduced in 1954 by A.F. Huxley and collaborators based on microscopy observations of muscle contraction [66, 65]. Muscle is composed of short segments called sarcomeres that are themselves made up of two types of overlapping filaments; actin and myosin. Actin filaments are attached to each end of the sarcomere and overlap with myosin filaments in the center. Myosin filaments are equipped with crossbridges that bind sites on the actin filament and act as molecular motors to generate force. Muscle contraction is achieved by crossbridges pulling the filaments to increase overlap, thus shortening the sarcomere.

Since the initial discovery of the sliding filament model, there has been significant effort towards constructing mathematical models of the sarcomere. Early models were based in mass action kinetics, yielding systems of coupled partial differential equations governing the dynamics. These models treated actin and myosin filaments as rigid so that any force generated by the sarcomere is simply the sum of independently acting crossbridges [64]. Subsequent experimental evidence demonstrated that filaments were in fact extensible [67, 168] motivating a new class of sarcomere models. Filament extensibility was explicitly incorporated into a novel a differential equation model in [110].

More recently, mechanistic simulations using Monte Carlo techniques have been used to simulate the sarcomere with locations of all crossbridges and binding sites tracked throughout the simulation. In contrast to PDE models, mechanistic Monte Carlo models explicitly track each crossbridge and section of filaments, providing a much more detailed description of dynamic behavior. Authors of [38] construct a spatially explicit model for interactions between single strands of actin and myosin. Subsequent work extended the spatially explicit to a multi-filament model with realistic lattice geometry [158] and considered the effect of scale variation in this geometry on the behavior of the sarcomere [176, 177]. The spatially explicit Monte Carlo technique is more expensive than the class of PDE models but allow for resolution of finer scale behavior that may be of importance. In particular, individual crossbridges acting as molecular motors explicitly influence local deformation of filaments and thus influence other crossbridges.

Scaling current sarcomere models to study the behavior of larger segments of muscle fiber requires coupling current single sarsomere models into a single simulation. Some work has explored this formulation in the PDE model case [22, 154] but scaling remains prohibitively computationally expensive. Building towards a scalable model, recent work has explored coupled systems of ordinary and partial differential equations governing thin filament activation and actin-myosin interactions, respectively [169]. However, large models of coupled sarcomeres remain challenging.

1.3 Organization and Contributions

The thesis is arranged into six chapters, including an introduction and discussion. In chapter 2 we develop a data-driven method for the discovery of partial differential equations. The third chapter focuses on neural network models for low dimensional dynamical systems forecasting in the case where data may be limited and noisy. The fourth chapter introduces a technique for smoothing data from a nonlinear dynamical system with governing equations known up to a set of parameters. Chapter five presents a novel data-driven method for modeling the half-sarcomere. Summaries of the contributions

within each chapter follow.

Chapter 2 introduces a framework for learning exact forms of partial differential equations via sparse regression. Sparse regression aims to learn exact governing equations for dynamical systems only from measurements of the state [18]. We assume that for a function u there is a partial differential equation $u_t = N(u)$ where the right hand side lies in the span of a set of candidate basis functions, $N \in \langle \{\theta_i\}_{i=1}^D \rangle$, called the library. For sufficiently large and appropriately chosen library, we also assume that the representation of N in $\{\theta_i\}_{i=1}^D$ is sparse. Then $u_t = N(u) = \sum_{i=1}^D \theta_i(u)\xi_i$ with $|supp(\xi)| = |\{j : \xi_j \neq 0\}| \ll D$. Finding coefficients ξ is an ill-conditioned problem even with abundant and clean data. We introduce a sparse regression method called sequential threshold ridge regression (STRidge) and demonstrate its effectiveness on a variety of canonical equations from mathematical physics. The method reliably recovers correct functional forms for a variety of PDEs including the Navier-Stokes momentum equation, even when data was corrupted by measurement noise. We also generalize this result to include partial differential equation with spatially or temporally varying coefficients.

Chapter 3 presents a method for interpolating governing equations of unknown dynamical systems with neural networks. Building on work in [53, 134] we train a neural network to interpolate a vector field by placing the network within a numerical timestepping scheme and training via enforcing accuracy in flow map predictions. The principle contribution of the work in this chapter is to explicitly consider measurement error in the flow map used to train the neural network. The neural network is trained simultaneously with estimates for measurement error at each timestep by minimizing a loss function based on joint posterior distribution of the dynamics and state estimate. By explicitly considering noise in the optimization, we are able to construct accurate forecasting models from datasets corrupted by considerable levels of noise and to separate the measurement error from the underlying state. We also discuss issues with over fitting and generalizability when using neural networks.

Chapter 4 introduces a novel approach for smoothing high-dimensional nonlinear dy-

namical systems. Rather than treating state estimates probabilistically, as in sequential Bayesian methods, we optimize over full length trajectories and use adherence to implicit Runge-Kutta schemes as a metric for plausibility. The resulting algorithm yields singular state estimates rather than distributions, thus taking on a more frequentest interpretation. However, the method differs from variational approaches in that state estimates need not exactly satisfy known governing equations. Like the Kalman filter, the proposed method weighs adherence to dynamics against fidelity to observed data. It assumes deterministic dynamics, making it less general than sequential Bayesian smoothers, but yields more accurate predictions on problems with deterministic dynamics. Furthermore, unknown parameters in the governing equations may be treated as variables in the optimization to get accurate parameter estimates even with signal to noise ratio less than 1. The method is robust to non-Gaussian noise, stiff, high-dimensional, and highly nonlinear dynamics, and even noise with non-zero mean.

Chapter 5 develops a data-driven model for the half sarcomere. The model is constructed to approximate the spatially explicit Monte Carlo technique developed in [38, 158] where crossbridge states affect neighbors through filament compliance. However, we do not explicitly track individual crossbridge states or locations. Average states of the sarcomere's crossbridges evolve through a three state time varying Markov chain model. Transition probabilities are functions of input parameters to the system as well as force generated by the sarcomere, which serves as a surrogate for compliance. Each transition probability is learned from data generated by the Monte Carlo simulation, thus allowing for the data-driven model to account for the macroscopic effects of filament compliance without explicitly considering spatial data. The data-driven model achieves accurate traces for force compared to the Monte Carlo simulation without the computational expense. The model is trained on periodically lengthened and activated sarcomere models and shown to be accurate on novel input regimes.

Bibliographic Notes

This dissertation is based upon the following publications and conference proceedings.

1. Samuel H. Rudy, C. David Williams, J. Nathan Kutz, and Thomas Daniel. A novel data-driven surrogate model of the half-sarcomere. *In preparation*
2. Samuel H. Rudy, Steven L. Brunton, and J. Nathan Kutz. Smoothing and parameter estimation by soft-adherence to governing equations. *arXiv preprint*, 2018 (Under review at *Journal of Computational Physics* special issue VSI: Machine Learning)
3. Samuel H. Rudy, J. Nathan Kutz, and Steven L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *arXiv preprint arXiv:1808.02578*, 2018 (Under review at *Journal of Computational Physics*)
4. Samuel H. Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-driven identification of parametric partial differential equations. SIAM J. Appl. Dyn. Syst., 18(2), 6435, 2017.
5. Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

Chapter 2

SPARSE IDENTIFICATION OF PARTIAL DIFFERENTIAL EQUATIONS

This chapter provides a detailed description of the PDE-FIND method, an algorithm used to discover governing partial differential equations from time series data collected on a spatial domain. The chapter is organized as follows: In Sec. 2.1 we discuss some background work in system identification and sparse regression techniques. Section 2.2 provides an overview of the PDE-FIND method for identifying constant coefficient equations, demonstrates the method on several canonical examples, and briefly discusses limitations of the method. Section 2.3 builds on the work in Sec. 2.2 to develop a method for identifying PDEs with non-constant coefficients. A brief discussion is given in Sec. 2.4.

2.1 Background

Research towards the automated inference of dynamical systems from data is not new [36]. Methods for extracting linear systems from time series data include the eigensystem realization algorithm (ERA) [73] and Dynamic Mode Decomposition (DMD) [136, 150, 165, 19, 85, 5]. Identification of nonlinear systems has, until very recently, relied on black box methods. These include neural networks [53, 140, 134], equation free methods [80, 81, 82], and others. There has also been considerable recent work towards data-driven approximation of the Koopman operator [108, 20, 109] via extensions of DMD [178], diffusion maps [51], delay-coordinates [16, 4, 41] and neural networks [179, 157, 172, 105, 116, 98, 140]. Many of these system identification techniques may be broadly cast in the the highly general NARMAX framework [25].

Many of the aforementioned algorithms rely on black box, or uninterpretable, repre-

sentations of the dynamics. A substantial step forward in nonlinear system identification occurred when scientists started to develop algorithms capable of symbolic regression, thus allowing for the discovery of true functional forms rather than interpolation. The use of genetic algorithms for nonlinear system identification [12, 151] allowed for the derivation of physical laws in the form of ordinary differential equations. Genetic algorithms are highly effective in learning complex functional forms but are slower computationally than simple regression. Sparsity promoting methods have been used previously in dynamical systems [143, 99, 21], and sparse regression has been leveraged to identify parsimonious ordinary differential equations from a large library of allowed functional forms [18, 26]. Much work building on the sparse regression framework has followed and includes inferring rational functions [103], the use of information criteria for model validation [104], constrained regression for conservation laws [92], model discovery with highly corrupt data [161], the learning of bifurcation parameters [147], stochastic dynamics [13], weak forms of the observed dynamics [145], and regression with small amounts of data [146, 76].

Sparse regression based methods for PDEs were first used in [139, 144]. These methods were demonstrated on a large class of PDEs and have the benefit of being highly interpretable, but struggle with numerical differentiation of noisy data. In Rudy *et al.* [139] the noise was addressed by testing with only a small degree of noise (large SNR), while in Schaeffer [144] noise was added to the time derivative after it was computed from clean data. The methods in [139, 144] were subsequently extended to parametric equation in [138].

Alternatively, Gaussian processes were used to determine linear PDEs [131] and nonlinear PDEs known up to a set of coefficients [130]. Using Gaussian process regression requires less data than sparse regression and naturally manages noise, but the method is only applicable to PDEs with a known structure. Reference [93] makes a substantial contribution by using neural networks to accurately learn partial differential equations with non constant coefficients. A neural network is constructed that mimics a forward Euler

timestepping scheme and the accuracy of potential models is evaluated based on their future state prediction accuracy. While seemingly more robust than sparse regression, the method in [93] does not penalize extraneous terms in the learned PDE and thus falls short of producing optimally parsimonious models. Furthermore, [93] only tests the method on a nonlinear problem using a relatively strong Ansatz. Neural networks were also used in [132] and [133] to solve and to estimate parameters in partial differential equations with known terms to a high degree of accuracy. However, similar to [130], it is assumed that the PDE is known up to a set of coefficients. A more sophisticated neural network approach was used in [129] to learn dynamics of systems with unknown terms. However, the approach in [129] does not give closed form representations of the dynamics and the resulting neural network model therefore does not give insights into the underlying physics.

This chapter is based on the sparse regression frameworks for constant coefficient and parametric PDEs developed in [139] and [138], respectively. The former is also closely related to the work in [144].

2.2 Constant Coefficient Equations

2.2.1 PDE-FIND

The PDE-FIND algorithm discussed in this work is a method for discovering the governing equation for a discretized dataset which we assume to be the solution to a PDE taking the form,

$$u_t = N(u, u_x, u_{xx}, \ldots, x, t, \mu) \tag{2.1}$$

where subscripts denote partial differentiation and μ represents parameters in the system. It is assumed that the function N may be expressed as a sum of a small number of terms, which is certainly the case for the PDEs considered here and/or widely used in practice. We denote $\mathbf{U}$ to be a matrix containing the values of u and $\mathbf{Q}$ as a matrix containing additional information that may be relevant, such as dependencies on the absolute value

of $|u|$, or another time varying function interacting with u.

While PDE-FIND does not have restrictions on the functional form of candidate terms in N, polynomial nonlinearities are common in many of the examples in this work and in many of the canonical models of mathematical physics. We may also consider data that is on a higher dimensional spatial domain, in which case we simply allow for derivatives with respect to each spatial dimension; u_x, u_y, u_{xy}, $u_x^2 u_{yy}$, etc.

PDE-FIND creates a large library of candidate terms that may appear in N, including nonlinearities and partial derivatives, and then selects a sparse subset of *active* terms from this list. In general, we always assume that the time evolution of a complex-valued function may depend on it's magnitude. The first step is to take the derivatives of the data with respect to time and each spatial dimension. Derivatives are taken either using finite differences for clean data, or when noise is added, with polynomial interpolation. The derivatives and the function itself are then combined into a matrix $\boldsymbol{\Theta}(\mathbf{U}, \mathbf{Q})$:

$$\boldsymbol{\Theta}(\mathbf{U}, \mathbf{Q}) = \begin{bmatrix} 1 & \mathbf{U} & \mathbf{U}^2 & \dots & \mathbf{Q} & \dots & \mathbf{U}_x & \mathbf{U}\mathbf{U}_x & \dots & \mathbf{Q}^2\mathbf{U}^3\mathbf{U}_{xxx} \end{bmatrix}. \tag{2.2}$$

Each column of $\boldsymbol{\Theta}$ contains all of the values of a particular candidate function across all of the grid points on which data is collected. Therefore, if we have data on an $n \times m$ grid (e.g. a 256×100 grid represents 256 spatial measurements at 100 time points) and have 50 candidate terms in the PDE, then $\boldsymbol{\Theta} \in \mathbb{C}^{256 \cdot 100 \times 50}$. We also take the time derivative to compute $\mathbf{U}_t$ and reshape it into a column vector just like we did the columns of $\boldsymbol{\Theta}$. This

gives a linear equation representing our PDE:

$$\mathbf{U}_t = \mathbf{\Theta}(\mathbf{U}, \mathbf{Q})\xi \tag{2.3a}$$

$$\begin{bmatrix} u_t(x_0,t_0) \\ u_t(x_1,t_0) \\ u_t(x_2,t_0) \\ \vdots \\ u_t(x_{n-1},t_m) \\ u_t(x_n,t_m) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u(x_0,t_0) & u_x(x_0,t_0) & \dots & u^5u_{xxx}(x_0,t_0) \\ 1 & u(x_1,t_0) & u_x(x_1,t_0) & \dots & u^5u_{xxx}(x_1,t_0) \\ 1 & u(x_2,t_0) & u_x(x_2,t_0) & \dots & u^5u_{xxx}(x_2,t_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u(x_{n-1},t_m) & u_x(x_{n-1},t_m) & \dots & u^5u_{xxx}(x_{n-1},t_m) \\ 1 & u(x_n,t_m) & u_x(x_n,t_m) & \dots & u^5u_{xxx}(x_n,t_m) \end{bmatrix}}_{\text{Example } \mathbf{\Theta} \text{ for real valued function in one spatial dimension}} \begin{bmatrix} \\ \\ \\ \\ \xi \\ \\ \\ \\ \\ \end{bmatrix} . \tag{2.3b}$$

Note that if we assume $\mathbf{\Theta}$ is an over complete library, meaning $\mathbf{\Theta}$ has a sufficiently rich column space that the dynamics will be in it's range, then the PDE should be well-represented by Eq. (2.3a) with a sparse vector of coefficients ξ. This amounts to picking enough candidate functions that the full PDE may be written as weighted sum. Each row of this linear system represents an observation of the dynamics at a particular point in time and space.

$$u_t(x,y) = \sum_j \Theta_j(u(x,t), q(x,t))\xi_j. \tag{2.4}$$

For an unbiased representation of the dynamics, we would simply solve the least squares problem for ξ. However, even with the only error coming from numerical round-off, the least-squares solution may be inaccurate. In particular, ξ will have predominantly nonzero values suggesting a PDE with every functional form contained in the library. Most importantly, for regression problems similar to (2.4), the least squares problem is poorly conditioned. Error in computing the derivatives (already an ill-conditioned problem with noise) will be magnified by numerical errors when inverting $\mathbf{\Theta}$. Furthermore,

measurement error also affects the least-squares solution. Thus if least squares is used, it can radically change the qualitative nature of the inferred dynamics. Instead, we utilize sparse regression to approximate a solution of

$$\xi = arg\,min_{\hat{\xi}} \|\boldsymbol{\Theta}\hat{\xi} - \mathbf{U}_t\|_2^2 + \lambda\|\hat{\xi}\|_0. \tag{2.5}$$

This assures that terms will only show up in the derived PDE if their effect on the error $\|\boldsymbol{\Theta}\hat{\xi} - \mathbf{U}_t\|$ outweigh their addition to $\|\hat{\xi}\|_0$. The ℓ^0 term makes this problem *np*-hard. In the next section we discuss methods for approximating solutions to Eq. (2.5).

Sparse Regression for PDE-FIND

A common technique for non-convex optimization problem such as (2.5) is to approximate with a similar convex problem [164]. The convex relaxation of the ℓ^0 optimization problem in (2.5) is

$$\xi = arg\,min_{\hat{\xi}} \|\boldsymbol{\Theta}\hat{\xi} - \mathbf{U}_t\|_2^2 + \lambda\|\hat{\xi}\|_1. \tag{2.6}$$

Minimization of the cost function given by equation (2.6) is known by the acronym LASSO, for least absolute shrinkage and selection operator [160]. LASSO is a commonly used technique in many sparse regression problems and, being convex, allows for rigorous analysis of convergence properties. However, LASSO tends to have difficulty finding a sparse basis when the data matrix $\boldsymbol{\Theta}$ has high correlations between columns, which is the case in many of our examples [184].

An alternative method for sparse regression was utilized in [18], called sequentially thresholded least squares (STLS). In STLS, a least squares predictor is obtained and a hard threshold is performed on the regression coefficients. The process is repeated recursively on the remaining nonzero coefficients. This is illustrated in algorithm 1 when $\lambda = 0$. STLS outperforms LASSO in some cases but does not avoid the challenge of correlation in the

data. Using a regularizer for the least squares problem can help avoid problems due to correlations. Ridge regression is an ℓ^2 regularized variation of least squares corresponding to the maximum a posteriori estimate using a Gaussian prior [114]. It is defined by,

$$\begin{aligned}\hat{\xi} &= argmin_{\xi}\|\mathbf{\Theta}\xi - \mathbf{U}_t\|_2^2 + \lambda\|\xi\|_2^2 \\ &= \left(\mathbf{\Theta}^T\mathbf{\Theta} + \lambda I\right)^{-1}\mathbf{\Theta}^T\mathbf{U}_t.\end{aligned} \tag{2.7}$$

We substitute ridge regression for least squares in STLS and call the resulting algorithm Sequential Threshold Ridge regression (STRidge), outlined in algorithm 1. Note that for $\lambda = 0$ this reduces to STLS. STRidge had the best empirical performance for PDE-FIND of any sparse regression algorithm tested in this work.

Since each value of the threshold tolerance will give a different level of sparsity in the final solution, we also used a separate method to find the best tolerance. Predictors are trained at varying tolerances and their performance on a holdout set, taking into account an ℓ^0 penalty, is used to find the best tolerance. We set the ℓ^0 penalty to be proportional to the condition number of $\mathbf{\Theta}$, enforcing sparsity in the case of highly correlated and ill-conditioned data. A multiplier of 10^{-3} was used based on empirical evidence. Our method for searching for the optimal tolerance is outlined in algorithm 2. Arguments passed into the search algorithm include $\mathbf{\Theta}$, $\mathbf{U}_t$, λ, and STR_iters which are passed directly to STRidge as well as d_tol and tol_iters. d_tol tells the search algorithm how large of a step to take while looking for the optimal tolerance and tol_iters indicates how many times the algorithm will refine it's guess at the best tolerance.

Algorithm 2 is designed such that the optimal tolerance will be found if the ℓ^0 penalized loss function, taken as a function of data and tol, is convex in tol. This is since it assumes the loss function will decrease until the optimal value of tolerance then start increasing again. This may not be the case for many examples but is shown to hold for many of the test problems considered here.

As a preprocessing step, each column of $\mathbf{\Theta}$ is normalized to unit variance. This is

especially useful if the function is not roughly $\mathcal{O}(1)$ so that higher powers are either very large or small. In all of the examples presented in this thesis, columns of Θ are normalized to unit length before solving for the sparse vector of coefficients ξ. A final prediction of ξ is obtained by regressing the non-normalized data onto the identified terms. The only instance in which this was less successful than STRidge without normalization was for identifying the advection diffusion equation from a biased random walk.

Several other methods exist for finding sparse solutions to least squares problem. Greedy algorithms have been shown to exhibit good performance on sparse optimization problems including PDE-FIND but in some cases were less reliable than STRidge [182]. While STRidge with normalization works well on almost all of the examples we tested (advection diffusion being the exception), we do not make the claim that it is optimal. The elastic-net algorithm, which has been shown to show advantages over LASSO, was also tested and found to be less effective for sparse regression than STRidge. [184]. If additional information regarding the PDE is known, for instance if we know one of the terms is nonzero, then this may be incorporated into the penalty on the coefficients.

One particular method stands out as a viable alternative to STRidge. The forward-backward greedy algorithm introduced by Zhang [182] originally introduced new active coefficients into a predictor based on the error they achieve training on the residual of previously picked terms. More recent work [159] has used a variation of this method where new active terms are selected based on their performance when trained with current active terms. This is slower than the original method but may be more robust to ill-conditioned problems. In particular, the authors of [159] found their method outperformed STRidge.

Algorithm 1 STRidge($\boldsymbol{\Theta}, \mathbf{U}_t, \lambda, tol$, iters)

$\hat{\xi} = arg\,min_{\xi} \|\boldsymbol{\Theta}\xi - \mathbf{U}_t\|_2^2 + \lambda\|\xi\|_2^2$ # ridge regression

$\mathcal{G} = \{j : |\hat{\xi}_j| \geq tol\}$ # select large coefficients

$\hat{\xi}[\sim \mathcal{G}\,] = 0$ # apply hard threshold

$\hat{\xi}[\mathcal{G}]$ = STRidge($\boldsymbol{\Theta}[:, \mathcal{G} =], \mathbf{U}_t, tol$, iters $- 1$) # recursive call with fewer coefficients

return $\hat{\xi}$

Algorithm 2 TrainSTRidge($\boldsymbol{\Theta}$, $\mathbf{U}_t$, λ, d_{tol}, tol_iters, STR_iters)

Split the data into training and testing sets

$$\left.\begin{array}{l} \boldsymbol{\Theta} \rightarrow [\boldsymbol{\Theta}^{train}, \boldsymbol{\Theta}^{test}] \\ \mathbf{U}_t \rightarrow [\mathbf{U}_t^{train}, \mathbf{U}_t^{test}] \end{array}\right\} \text{ 80/20 split}$$

Set an appropriate ℓ^0-penalty. The following worked well empirically

$\eta = 10^{-3}\kappa(\boldsymbol{\Theta})$

Get a baseline predictor

$\xi_{best} = (\boldsymbol{\Theta}^{train})^{-1}\,\mathbf{U}_t^{train}$

error$_{best}$ $= \|\boldsymbol{\Theta}^{test}\xi_{best} - \mathbf{U}_t^{test}\|_2^2 + \eta\|\xi_{best}\|_0$

Now search through values of tolerance to find the best predictor

$tol = d_{tol}$

for $iter = 1, \ldots,$ tol_iters:

    # Train and evaluate performance

    ξ =STRidge($\boldsymbol{\Theta}^{train}, \mathbf{U}_t^{train}, \lambda, tol,$ STR_iters)

    error$= \|\boldsymbol{\Theta}^{test}\xi - \mathbf{U}_t^{test}\|_2^2 + \eta\|\xi\|_0$

    if error$\leq$ error$_{best}$: # Is the error still dropping?

        error$_{best}$ =error

        $\xi_{best} = \xi$

        $tol = tol + d_{tol}$

    else: # Or is tolerance too high?

        $tol = \max([0, tol - 2d_{tol}])$

        $d_{tol} = \frac{2d_{tol}}{\text{tol\_iters} - iter}$

        $tol = tol + d_{tol}$

return ξ_{best}

Numerical Differentiation

When using PDE-FIND on clean data from numerical simulations we take derivatives with second order finite differences [88, 84]. Numerically differentiating noisy data is considerably more challenging. If one uses finite difference techniques on a grid with spacing $\mathcal{O}(h)$ and noise with amplitude $\mathcal{O}(\epsilon)$ then the d^{th} derivative will have noise approximately of $\mathcal{O}(\frac{\epsilon}{h^d})$, which will result in numerical derivatives being dominated by the effects of noise. We consider four other methods for numerical differentiation when there was noise added to the solution.

A simple variation of finite differences is to use a smoothing technique on the noisy data such as interpolation with a spline or convolving with a smoothing kernel. We tried the latter, using a Gaussian smoothing kernel on noisy data prior to taking derivatives with finite differences. While the derivatives we obtained seemed to be free of noise, they were sufficiently biased to create problems with identifying the dynamics. Convolving with a Gaussian has the effect of nearly eliminating higher frequency components of a signal, as it is equivalent to multiplying the spectral representation by a Gaussian. This smooths out sharp inflections in the curve.

Tikhonov differentiation finds a numerical derivative $\hat{f}'$ for a function f by balancing the closeness of the integral of $\hat{f}'$ to f with the smoothness of $\hat{f}'$ [83]. The continuous problem is given by,

$$\hat{f}' = argmin_g \left\| \int_{x_0}^{x} g(s)\, ds - (f(x) - f(x_0)) \right\|_2^2 + \lambda \left\| \frac{dg}{dx} \right\|_2^2 \tag{2.8}$$

where g is taken to be in an appropriate class of functions and $\|\cdot\|_2^2$ is evaluated over an interval where the derivative is approximated. The discrete version of (2.8) is given by,

$$\hat{f}' = argmin_g \|Ag - (f - f_0)\|_2^2 + \lambda \|Dg\|_2^2, \tag{2.9}$$

where A is a trapezoidal approximation to the integral and D is a finite difference approx-

imation to the derivative. Equation (2.9) has closed form solution given by,

$$\hat{f}' = \left(A^T A + \lambda D^T D\right)^{-1} A^T (f - f_0).$$

Tikhonov differentiation, similar to using a smoothing kernel, results in a far smoother numerical derivative than finite differences but also introduces a small amount of bias, particularly for functions with large higher order derivatives.

When the data is on a periodic domain, the best method for taking the dth derivative may be via the discrete Fourier transform and multiplication by $(ik)^d$ in the frequency domain. To combat the effects of noise one could use a filter in the frequency domain. Doing so, however, would require a principled method for deciding exactly how to threshold high frequency terms without distorting the shape of the curve. Furthermore, we cannot always assume a period spatial domain or use the Fourier transform to differentiate with respect to time. Spectral differentiation was not implemented in the examples in this manuscript. We suspect this would be a promising tool for data considered on a periodic domain, or a sufficiently wide domain that the data goes to zero at the boundaries.

The method we found to be the easiest to implement and most reliable for noisy data was polynomial interpolation [83]. For each data point where we compute a derivative, a polynomial of degree p was fit to greater than p points and derivatives of the polynomial were taken to approximate those of the numerical data. Points close to the boundaries where it was difficult to fit a polynomial were not used in the regression. This method is far from perfect; data close to the boundaries was difficult to differentiate and the result of PDE-FIND depended strongly on the degree of polynomial and number of points used to fit it. For a more principled but involved approach to polynomial differentiation, see [15].

Subsampling data

For large datasets such as those with more than one spatial dimension, PDE-FIND can be used on subsampled data. We randomly select a set of spatial points and evenly sample

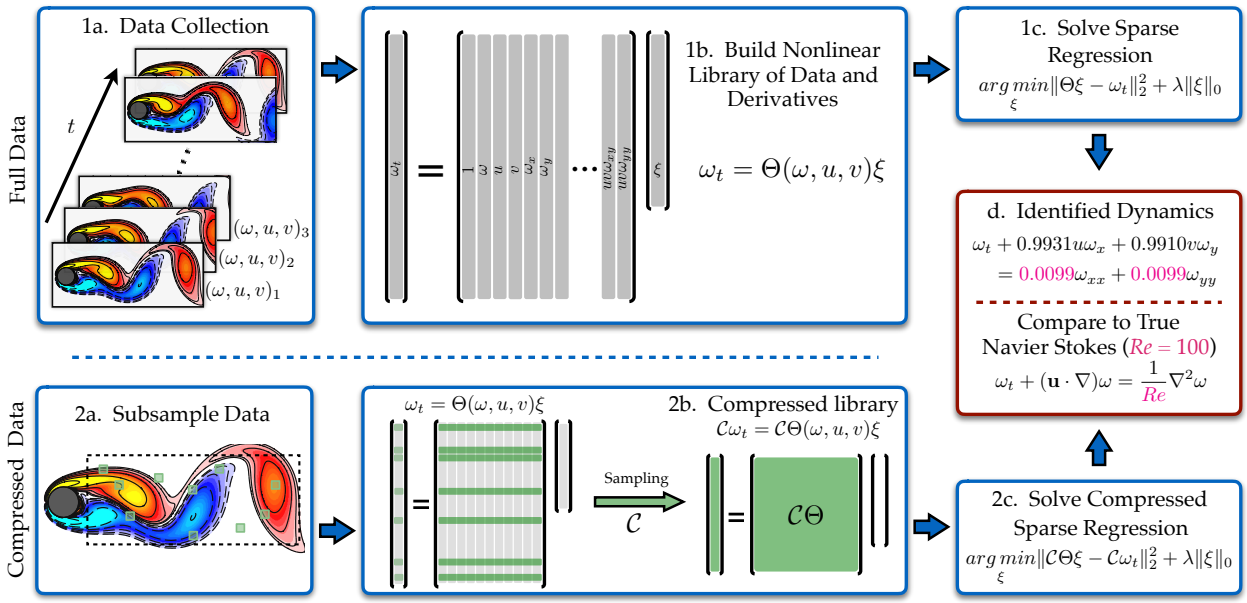


Figure 2.1: Steps in the PDE functional identification of nonlinear dynamics (PDE-FIND) algorithm, applied to infer the Navier-Stokes equation from data. **1a.** Data is collected as snapshots of a solution to a PDE. **1b.** Numerical derivatives are taken and data is compiled into a large matrix $\mathbf{\Theta}$, incorporating candidate terms for the PDE. **1c.** Sparse regressions is used to identify active terms in the PDE. **2a.** For large datasets, sparse sampling may be used to reduce the size of the problem. **2b.** Subsampling the dataset is equivalent to taking a subset of rows from the linear system in (2.3). **2c.** An identical sparse regression problem is formed but with fewer rows. **d.** Active terms in ξ are synthesized into a PDE.

the data in time at a lower frequency than data is collected, resulting in the use of only a fraction of the dataset. Mathematically, this amounts to ignoring a fraction of the rows in the linear system $\mathbf{U}_t = \mathbf{\Theta}(\mathbf{U}, \mathbf{Q})\xi$ as illustrated in Figure 2.1 panels 2a and 2b.

Though we only use a small fraction of the spatial points in the linear system, nearby points are needed to evaluate the derivative terms in the library. The derivatives were computed via polynomial interpolation, using a small number of points close to the point in question to fit to a polynomial. Therefore, while subsampling uses only a small fraction of the points in the regression, we are also using local information around each of the measurements.

PDE-FIND for a Fokker-Planck equation

A trivial result in stochastic processes is that the density function of a Brownian motion evolves according to the diffusion equation [48]. Biasing the Brownian motion introduces an advection term. As a test problem for the proposed methodology on automated PDE discovery, we consider a simple methodology for studying the evolution of the density of a random process.

Under a fairly nonrestrictive set of assumptions regarding a stochastic trajectory, a PDE may also be derived for the distribution function of the future position from just the single single trajectory, also known as the Fokker-Planck equation [48]. Let $X(t)$ be the position of a particle undergoing a random walk. We assume the trajectory follows the rule that the displacement of the particle over an interval of time t, $X(t+\tau) - X(\tau)$, may be predicted according to a probability distribution which at time zero is a point mass and which does not depend on τ or $X(\tau)$. That is,

$$X(t+\tau) - X(\tau) \sim u(x,t) \text{ where } u(x,0) = \delta(x). \tag{2.10}$$

The important point here is that we assume enough to justify splitting the time series into pieces that all follow the same PDE, independent of their location in time and space. PDE-FIND looks for the relation $u_t = N(u, u_x, \ldots)$ by approximating the distribution function u using histograms.

We start with a single time series consisting of evenly spaced measurements of the stochastic trajectory, $X = (X_0, X_1, \ldots, X_n)$. This time series is split into many shorter series, H_j of length p (we used p = 5):

$$H_j = (X_{j+1}, X_{j+2}, \ldots, X_{j+p}) - X_j \tag{2.11a}$$

$$= (H_j^1, H_j^2, \ldots, H_j^p) \qquad \text{for } j = 1, \ldots, n-p. \tag{2.11b}$$

For each of the p timesteps, it is possible to build a histogram across all of the H_j time

series. These binned histograms approximate the discretized version of our probability density u on a grid of size $n \times p$. We may then compute spatial and temporal derivatives for use in PDE-FIND.

When using histograms to approximate the density function it is important to choose values of n (number of bins) and p (number of time steps) that are sufficiently high to be able to accurately differentiate the density function, but not so high that the volume of data collection is not sufficient to resolve the density function at that granularity. For example, in the diffusion example we expect the density at each time step to be a Gaussian. If n is too low, then we will not be able to compute spatial derivatives well but if it too high then we will not have enough data to adequately approximate the density in each bin. When choosing p, we need sufficiently many time steps to evaluate temporal derivatives but since the density function spreads out we cannot choose p too high or else we will not be able to approximate the very wide distribution that results.

Filtering noise via singular value decomposition

For some datasets we may be able to denoise via exploiting low dimensional structures in the data. The singular value decomposition [163] is utilized to discover low-energy directions in the data corresponding to additive noise. Applied to spatial-temporal datasets, this is often referred to as the proper orthogonal decomposition (POD). Modes with larger singular values correspond to recurrent structures in the data. Typically, only a few of these modes are required to reconstruct the dynamics with low error [9, 125, 18]. Principled truncations methods for denoising via the SVD for square matrices are explained in detail in [49].

Adding noise to a spatio-temporal dataset erases features corresponding to low singular values while leaving coherent structures largely unaffected. We truncate the SVD according the optimal hard threshold criterion [49]. The result is a low dimensional approximation of the noisy dataset that we assume to be less noisy than the original while

maintaining all of the important dynamics. We employed an SVD filter for noise in the examples for Navier-Stokes and the reaction diffusion equation. Each equation was identified from a low dimensional subspace recovered from its most important singular vectors.

2.2.2 Examples

This section contains a number of test cases on which we applied PDE-FIND. In each case that the dynamics were specified in an Eulerian frame of reference, PDE-FIND was used both on clean data using finite differences to take derivatives, and on noisy data using polynomial interpolation. For the artificial noise, we used white noise with magnitude equal to 1% of the standard deviation of the solution function. For complex functions, both real and imaginary noise was added, each having magnitude $1/\sqrt{2}\%$ of the real and complex components of the function, respectively. Details on numerical methods for creating training data are also given. Where not specified, the numerical solution was obtained via spectral differentiation and a Runge-Kutta-45 ODE solver. Table 2.1 summarizes the results, including error in identifying each PDE with and without noise, and the discretization used in each case.

The KdV Equation

The KdV equation is an asymptotic simplification of Euler equations used to model waves in shallow water. It can also be viewed as Burgers' equation with an added dispersive term. Traveling waves in a solution to the KdV equation behave linearly when alone but exhibit nonlinear interactions. However, any solution with waves at multiple amplitudes will exhibit nonlinear behavior irregardless of interaction due to the amplitude dependence of wave speed.

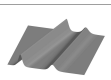
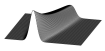
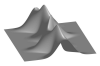
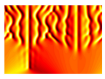
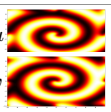
PDE	Form	Error (no noise, noise)	Discretization
 KdV	$u_t + 6uu_x + u_{xxx} = 0$	$1\%\pm0.2\%, 7\%\pm5\%$	$x\in[-30,30]$, n=512, $t\in[0,20]$, m=201
 Burgers	$u_t + uu_x - \epsilon u_{xx} = 0$	$0.15\%\pm0.06\%, 0.8\%\pm0.6\%$	$x\in[-8,8]$, n=256, $t\in[0,10]$, m=101
 Schrödinger	$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0$	$0.25\%\pm0.01\%, 10\%\pm7\%$	$x\in[-7.5,7.5]$, n=512, $t\in[0,10]$, m=401
 NLS	$iu_t + \frac{1}{2}u_{xx} + \lvert u\rvert^2u = 0$	$0.05\%\pm0.01\%, 3\%\pm1\%$	$x\in[-5,5]$, n=512, $t\in[0,\pi]$, m=501
 KS	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$	$1.3\%\pm1.3\%, 52\%\pm1.4\%$	$x\in[0,100]$, n=1024, $t\in[0,100]$, m=251
u  v Reaction Diffusion	$u_t = 0.1\nabla^2u + \lambda(A)u - \omega(A)v$ $v_t = 0.1\nabla^2v + \omega(A)u + \lambda(A)v$ $A^2 = u^2+v^2, \omega = -\beta A^2, \lambda = 1-A^2$	$0.02\% \pm 0.01\%, 3.8\% \pm 2.4\%$	$x, y\in[-10,10]$, n=256, $t\in[0,10]$, m=201 subsample 1.14%
 Navier-Stokes	$\omega_t + (\mathbf{u}\cdot\nabla)\omega = \frac{1}{Re}\nabla^2\omega$	$1\% \pm 0.2\%$, $7\% \pm 6\%$	$x\in[0,9]$, n_x=449, $y\in[0,4]$, n_y=199, $t\in[0,30]$, m=151, subsample 2.22%

Table 2.1: Summary of regression results for a wide range of canonical models of mathematical physics. In each example, the correct model structure is identified using PDE-FIND. The spatial and temporal sampling used for the regression is given along with the error produced in the parameters of the model for both no noise and 1% noise. In the reaction-diffusion (RD) system, 0.5% noise is used. For Navier-Stokes and Reaction Diffusion, the percent of data used in subsampling is also given.

Correct PDE	$u_t + 6uu_x + u_{xxx} = 0$
Identified PDE (clean data)	$u_t + 5.956uu_x + 0.988u_{xxx} = 0$
Identified PDE (1% noise)	$u_t + 6.152uu_x + 1.124u_{xxx} = 0$

Table 2.2: Summary of PDE-FIND for identifying the KdV equation.

Two soliton solution of KdV

PDE-FIND was tested on a very simple solution to the KdV equation having two non-interacting traveling waves with different amplitudes. The two soliton solution to the KdV equation was created using a spectral method with 512 spatial points and 200 timesteps. The initial condition was a superposition of functions of the form in Eq. (2.12) with offset centers and different amplitudes (Figure 2.2). Results are summarized in Table 2.2.

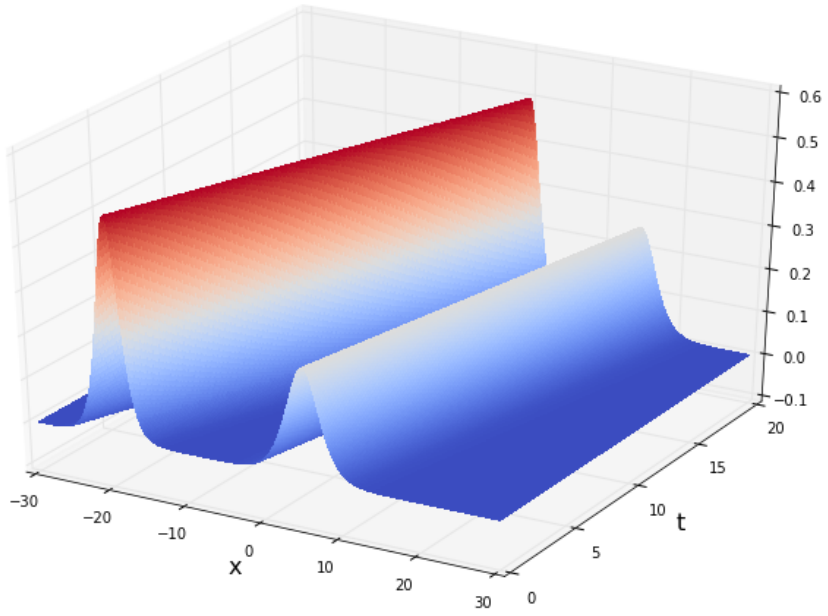


Figure 2.2: The numerical solution to the KdV equation.

Disambiguating linear and nonlinear waves

Some of the discretized solutions studied may be solutions to more than one PDE, even within the span of the candidate functions used in PDE-FIND. An example is the single soliton solution to the KdV equation, which is a hyperbolic secant squared solution

$$u(x,t) = \frac{c}{2}\text{sech}^2\left(\frac{\sqrt{c}}{2}(x - ct - a)\right) \tag{2.12}$$

that travels with a speed c. However, the one-way wave equation $u_t + cu_x = 0$ generically admits solutions of the form $u = f(x - ct)$ where the function f is prescribed by the initial data. If the initial data was a hyperbolic secant squared, then the solution for both the KdV and one-way wave equation admit the same traveling wave solution. Application of our sparse regression framework would select the one-way wave equation since it is the more sparse of the two PDEs. Ultimately, a technique must be capable of disambiguating between these two PDEs. This can be done by providing different initial (amplitude) conditions. For the one-way wave equation, the waves would still travel with speed c whereas for the KdV the speed c would be amplitude dependent.

We constructed two solutions of this form having c equal to 1 or 5 on grids with 256 spatial points and 50 timesteps. In this case noise was not added to the solution so derivatives were taken via finite differences. When using a single traveling wave in PDE-FIND, we identified the advection equation with corresponding c. We also tried training a sparse predictor using both datasets by stacking them on top of one another in a linear system:

$$\mathbf{U}_t = \begin{bmatrix} \mathbf{U}_t^{c=1} \\ \mathbf{U}_t^{c=5} \end{bmatrix} = \begin{bmatrix} \mathbf{\Theta}^{c=1} \\ \mathbf{\Theta}^{c=5} \end{bmatrix} \xi = \mathbf{\Theta}\xi.$$

Any solution of this linear system must represent a nonlinear PDE, since an increase in amplitude results in a faster wave. When PDE-FIND is used for both traveling waves simultaneously we identify the KdV equation.

Correct PDE	$u_t + uu_x = u_{xx}$
Identified PDE (clean data)	$u_t + 1.001uu_x = 0.100u_{xx}$
Identified PDE (1% noise)	$u_t + 1.010uu_x = 0.103u_{xx}$

Table 2.3: Summary of PDE-FIND for identifying Burgers' equation.

Burgers' equation

Burgers' equation is derived from the Navier-Stokes equations for the velocity field by dropping the pressure gradient term. Despite it's relation to the much more complicated Navier-Stokes equations, Burgers' equation does not exhibit turbulent behavior and may even be linearized through the Cole-Hopf transform [61, 31]. PDE-FIND was tested on a solution to Burgers' equation with a Gaussian initial condition, propagating into a traveling wave. Unlike many solutions to the inviscid Burgers' equation ($u_t + uu_x = 0$), the dissipative term u_{xx} prevents a shock from forming. This is important for our being able to identify the PDE since PDE-FIND relies on differentiable solutions. See Figure 2.3 for numerical solution. Results are summarized in Table 2.3.

Quantum Harmonic Oscillator

The quantum harmonic oscillator is Schrödinger's equation with a parabolic potential. The harmonic oscillator can be solved explicitly using Gauss-Hermite polynomials. It models the time evolution of the wave function associated with a particle in the parabolic potential, and at any point will give the distribution function of the particles location via taking the squared magnitude. It is important to note that measuring this function in practice is impossible as one can only observe a particles location and not it's distribution function. Further, even if some statistical distribution is gathered from many experiments, it will lack any information regarding the complex phase of the wave function. Therefore, this particular example is merely theoretical, though it does show the effectiveness of

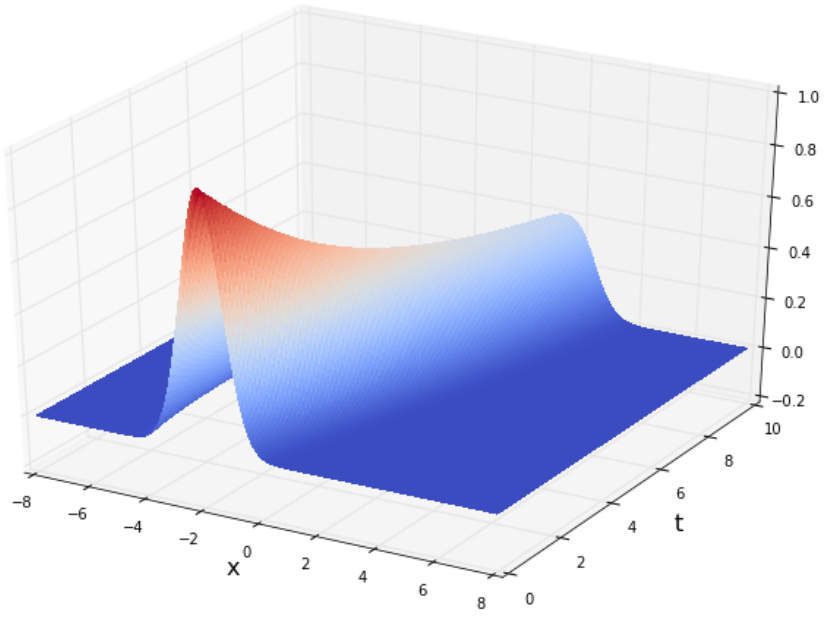


Figure 2.3: The numerical solution to the Burgers' equation.

PDE-FIND for complex valued functions. The magnitude of the solution used is shown in Figure 2.4.

Candidate functions for the PDE were given as quadratic polynomials in u, $|u|$, and $q = x^2/2$ multiplying either a constant or a derivative of u up to the third derivative; note that in the quantum mechanics literature, the wave function u is often denoted ψ and the potential q is often denoted $V(x)$. Though the magnitude $|u|$ does not appear in the correct PDE, we naively assume that any complex function may follow a PDE involving it's absolute value (as in the nonlinear Schrödinger equation) and allow for our sparse regression to show that it does not. Results of the PDE-FIND algorithm are shown in Table 2.4.

Correct PDE	$u_t = 0.5iu_{xx} - iuV$
Identified PDE (clean data)	$u_t = 0.499iu_{xx} - 0.997iuV$
Identified PDE (1% noise)	$u_t = 0.417iu_{xx} - 1.027iuV$

Table 2.4: Summary of PDE-FIND for identifying the Schrödinger equation.

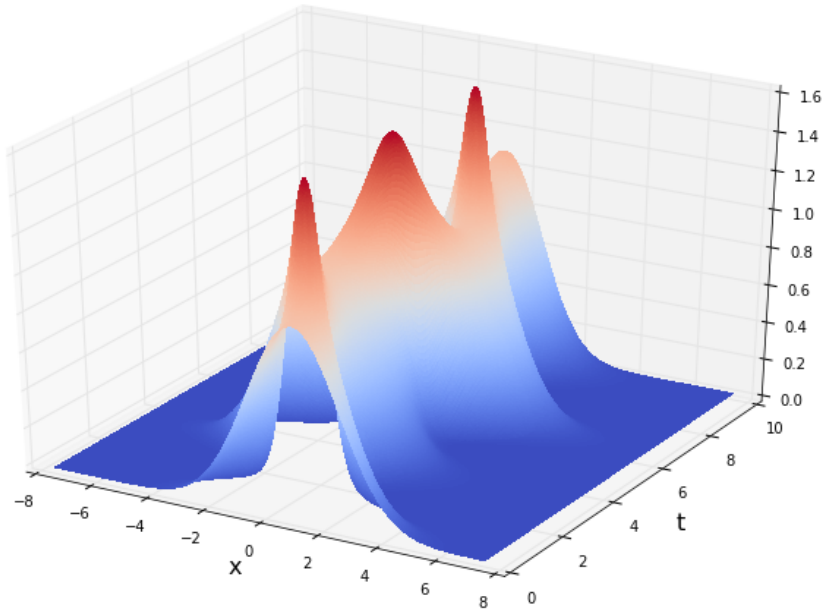


Figure 2.4: The magnitude of the numerical solution to the Schrödinger's equation.

Nonlinear Schrödinger equation

The Nonlinear Schrödinger equation is used to study nonlinear wave propagation in optical fibers and/or waveguides, Bose-Einstein condensates (BECs), and plasma waves. In optics, the nonlinear term arises from the intensity dependent index of refraction of a given material. Similarly, the nonlinear term for BECs is a result of the mean-field interactions of an interacting, N-body system. We use PDE-FIND to identify the equation from a breather solution with Gaussian initial condition. Since the function is complex val-

Correct PDE	$u_t = 0.5iu_{xx} + i\|u\|^2u$
Identified PDE (clean data)	$u_t = 0.500iu_{xx} + 1.000i\|u\|^2u$
Identified PDE (1% noise)	$u_t = 0.479iu_{xx} + 0.982i\|u\|^2u$

Table 2.5: Summary of PDE-FIND for identifying the Nonlinear Schrödinger equation.

ued, we consider candidate functions that have terms depending on the magnitude of the solution, $|u|$, which is of course necessary for the correct identification of the dynamics.

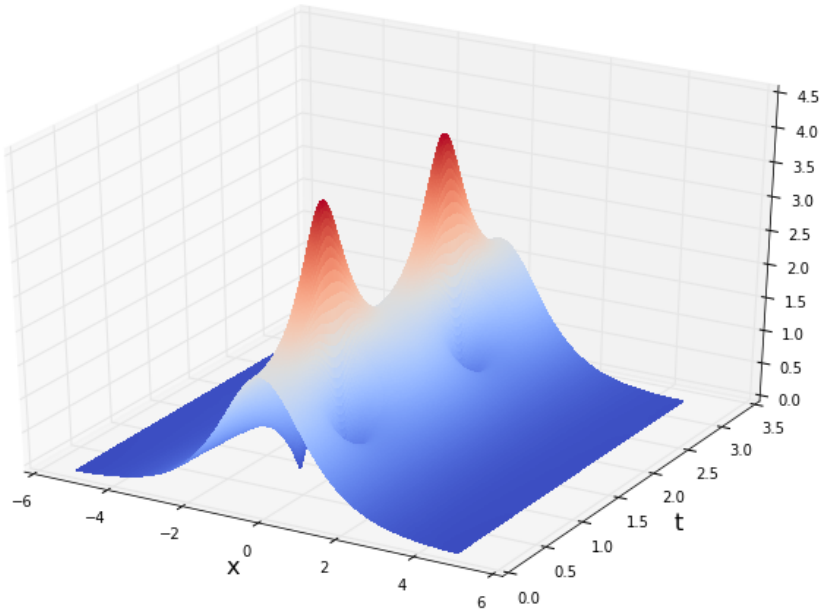


Figure 2.5: The magnitude of the numerical solution to the nonlinear Schrödinger's equation.

Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky (KS) equation has been independently derived in the context of several extended physical systems driven far from equilibrium by intrinsic instabili-

Correct PDE	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$
Identified PDE (clean data)	$u_t + 0.984uu_x + 0.994u_{xx} + 0.999u_{xxxx} = 0$
Identified PDE (1% noise)	$u_t + 0.459uu_x + 0.481u_{xx} + 0.492u_{xxxx} = 0$

Table 2.6: Summary of PDE-FIND for identifying the Kuramoto Sivashinksy equation.

ties [35]. It has been posited as a model for instabilities of dissipative trapped ion modes in plasmas, laminar flame fronts, phase dynamics in reaction-diffusion systems, and fluctuations in fluid films on inclines. More broadly, it is a canonical model of a pattern forming system with spatio-temporal chaotic behavior. Like the Burgers' equation, the KS equation provides a diffusive regularization of the nonlinear wave-breaking dynamics given by $u_t + uu_x = 0$. In this case, the stabilizing regularization is accomplished by a fourth-order diffusion term since the second order diffusion corresponds to long-wavelength instabilities, i.e. it is the backwards diffusion equation, which leads to blowup. This diffusive regularization is much like the Swift-Hohenberg equation.

We simulated the Kuramoto-Sivashinsky equation using a spectral method [162] with 1024 spatial grid points for 251 timesteps. While a courser solution sufficed for identifying the dynamics with clean data, the fine grid was needed to accurately identify the dynamics with noise. Results of the PDE-FIND algorithm are shown in Table 2.6. Note that though the correct terms were identified for the PDE, parameter error for the noisy data was very high. All coefficients were underestimated by roughly 50%. For lesser amounts of noise a similar undershooting of coefficients was observed.

While the high error realized in identifying the Kuramoto-Sivashinsky equation might be alarming, we note that the same equation has proved difficult for numerous other machine learning based system identification methods. The recent physics informed neural network framework was able to learn the Kuramoto-Sivashinsky dynamics in the non-chaotic regime but failed or the dataset considered here [129].

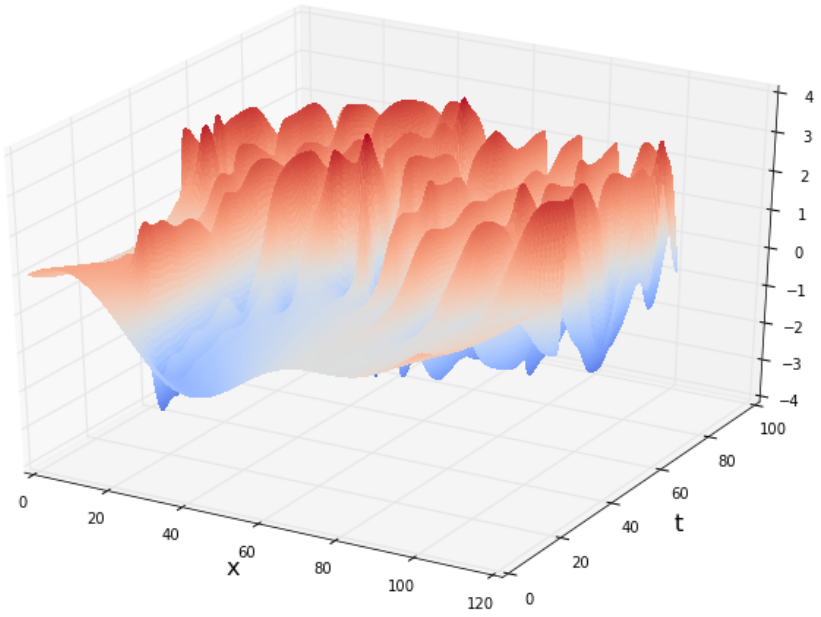


Figure 2.6: The numerical solution to the Kuramoto-Sivashinsky equation.

Reaction diffusion equation

Reaction diffusion systems have been widely used in mathematical physics to study pattern forming systems [35]. The dynamics produced by reaction diffusion systems can encompass most patterns observed in nature including target patterns, spiral waves, rolls, zig-zags, etc. As such, they have been the subject of intense research over many decades. Interestingly, most reaction diffusion systems are qualitatively derived and lack connection to first principles modeling. One class of reaction diffusion system commonly considered is the $\lambda - \omega$ system of the form:

$$u_t = 0.1\nabla^2 u + \lambda(A)u - \omega(A)v \tag{2.13a}$$

$$v_t = 0.1\nabla^2 v + \omega(A)u + \lambda(A)v \tag{2.13b}$$

$$A = u^2 + v^2, \omega = -\beta A^2, \lambda = 1 - A^2. \tag{2.13c}$$

Correct PDE	$u_t = 0.1u_{xx} + 0.1u_{yy} - uv^2 - u^3 + v^3 + u^2v + u$ $v_t = 0.1v_{xx} + 0.1v_{yy} + v - uv^2 - u^3 - v^3 - u^2v$
Identified PDE (clean data)	$u_t = 0.100u_{xx} + 0.100u_{yy} - 1.000uv^2 - 1.000u^3 + 1.000v^3 + 1.000u^2v + 1.000u$ $v_t = 0.100v_{xx} + 0.100v_{yy} + 1.000v - 1.000uv^2 - 1.000u^3 - 1.000v^3 - 1.000u^2v$
Identified PDE (0.5% noise)	$u_t = 0.095u_{xx} + 0.095u_{yy} - 0.945uv^2 - 0.945u^3 + 1.000v^3 + 1.000u^2v + 0.945u$ $v_t = 0.095v_{xx} + 0.095v_{yy} + 0.946v - 1.000uv^2 - 1.000u^3 - 0.946v^3 - 0.946u^2v$

Table 2.7: Summary of PDE-FIND for identifying reaction diffusion equation.

This particular reaction diffusion equation exhibits spiral waves on a 2D domain with periodic boundaries. To denoise the solution, we used the proper orthogonal decomposition and truncated to a lower dimensional representation based on the apparent Pareto front in the singular values. Both u and v were projected onto the 15 dimensional subspace defined by the vectors corresponding to there largest singular values.

Denoising via the SVD tends to work best when the solution is inherently low dimensional in a stationary frame, which is not the case for a traveling wave. Denoising was therefore less effective here than for a solution with more stationary features. The reaction diffusion equation was the only one of our examples for which we were unable to accurately identify the model with 1% noise and instead used 0.5%. Raising the noise to 1%, we were almost able to identify the correct PDE but the equation for U_t had an added linear dependence on v and v_t on u. In our identification of the reaction diffusion system, we subsampled 5000 spatial points and used 30 time points at each, resulting in 150,000 points or $\sim$1.14% of the dataset.

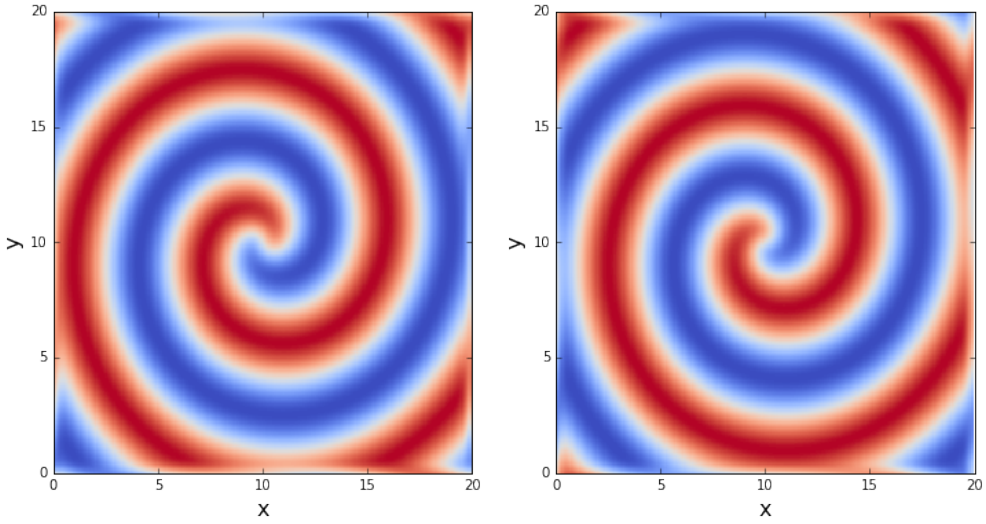


Figure 2.7: The numerical solution to the reaction diffusion equation.

Navier-Stokes

The Navier-Stokes equations describing the two-dimensional fluid flow past a circular cylinder at Reynolds number 100 are simulated using the Immersed Boundary Projection Method (IBPM) [155, 32][1]. This approach utilizes a multi-domain scheme with four nested domains, each successive grid being twice as large as the previous. Length and time are nondimensionalized so that the cylinder has unit diameter and the flow has unit velocity. Data is collected on the finest domain with dimensions 9×4 at a grid resolution of 449×199. The flow solver uses a 3rd-order Runge Kutta integration scheme with a time step of $\Delta t = 0.02$, which has been verified to yield well-resolved and converged flow fields; flow snapshots are saved every $10\Delta t = 0.2$. At this Reynolds number, the flow past a cylinder is characterized by periodic laminar vortex shedding, providing a rich, yet simple prototypical dynamical system to explore high-dimensional fluid data [115].

We identified the time dependence for the vorticity of the flow field as a function of vorticity and velocity. Candidate functions were taken to be polynomial terms of the vor-

[1]We use the C++ code available at: https://github.com/cwrowley/ibpm.

Correct PDE	$\omega_t = 0.01\omega_{xx} + 0.01\omega_{yy} - u\omega_x - v\omega_y$
Identified PDE (clean data)	$\omega_t = 0.00988\omega_{xx} + 0.00990\omega_{yy} - 0.990u\omega_x - 0.987v\omega_y$
Identified PDE (1% noise)	$\omega_t = 0.0107\omega_{xx} + 0.0083\omega_{yy} - 0.988u\omega_x - 0.983v\omega_y$

Table 2.8: Summary of PDE-FIND for identifying the Navier-Stokes equation.

ticity ω and the x and y coordinates of the velocity field up to second degree, multiplied by derivatives of the vorticity up to second order.

Since the data collected for Navier-Stokes exhibited coherent modes which dominated the long term behavior of the solution, the singular values of the dataset fell off rapidly, indicating that the majority of the behavior could be characterized with only a few dominant modes. This was especially useful in the case of noisy data since we were able to use the SVD to denoise the data more effectively than for the reaction diffusion system.

Like for the reaction diffusion problem, we also subsampled for Navier-Stokes. The dataset we used for the Navier-Stokes equations consisted of roughly 13.5 million data points. Constructing a data matrix for this many points across an overcomplete library of candidate functions could quickly become intractable, both for the time required to take derivatives as well as the space needed to store and work with that large of a matrix. We sampled 5000 spatial locations within a region of the domain downstream from the cylinder (see Figure 2.8) and collected the data and derivatives at these points for 60 different times, resulting in a matrix with 2.22% of the rows that would be present in the full dataset.

Location of sampling points may play a critical role in the Navier-Stokes example. The linear diffusion term scales as the inverse of the Reynolds number. Even at $Re = 100$, this coefficient is small. If sampling points are taken in areas of the domain where flow is relatively smooth then columns in $\boldsymbol{\Theta}$ corresponding to $\Delta\omega$ will be small. Column normalization of $\boldsymbol{\Theta}$ will therefore shrink the coefficients for the diffusion term even smaller, making it likely that they will be removed through the thresholding algorithm. Indeed,

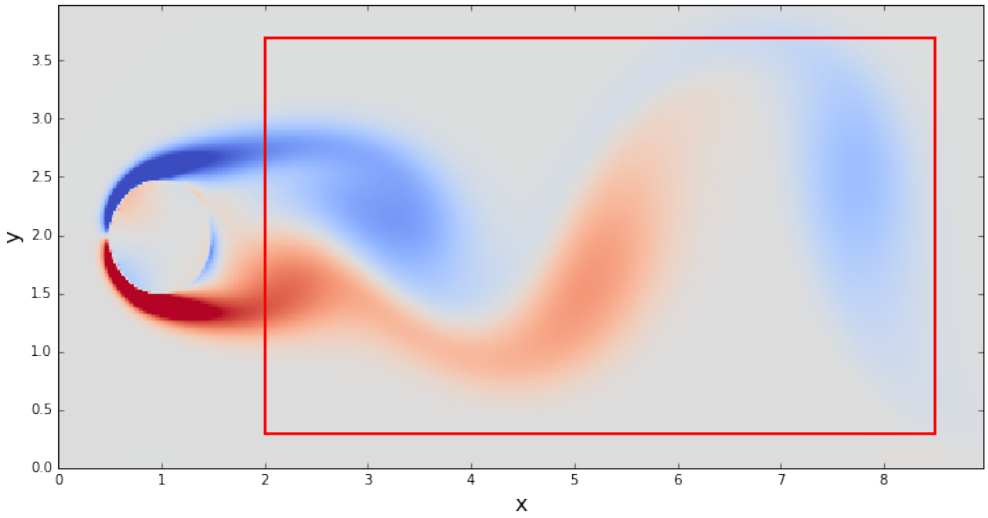


Figure 2.8: A single snapshot of the vorticity field is illustrated for the fluid flow past a cylinder. The sampling region is outlined in red.

for smaller numbers of sample points, we observed the ω_{xx} term was often set to zero.

Diffusion from a random walk

PDE-FIND was used to identify the long celebrated relation between Brownian motion and the diffusion equation. The Fokker-Planck equation associated with a particle's position, for Brownian motion where $x(t+dt) \sim \mathcal{N}(x(t), dt)$ is $u_t = 0.5u_{xx}$. Using a single trajectory of Brownian motion we are able to consistently derive the diffusion equation from a small number of observations.

We simulated Brownian motion at evenly space time points by adding a normally distributed random variable with variance dt to the time series. An example is shown in Figure 2.9a. Histograms of the particles displacement were taken using a number of bins that balanced resolution without overfitting the amount of data. Small changes in the number of bins, especially for longer time series, did not have a large effect on the result of PDE-FIND. Figure 2.10 shows an example of the histograms used to approximate the distribution function for a time series of length 10^6.

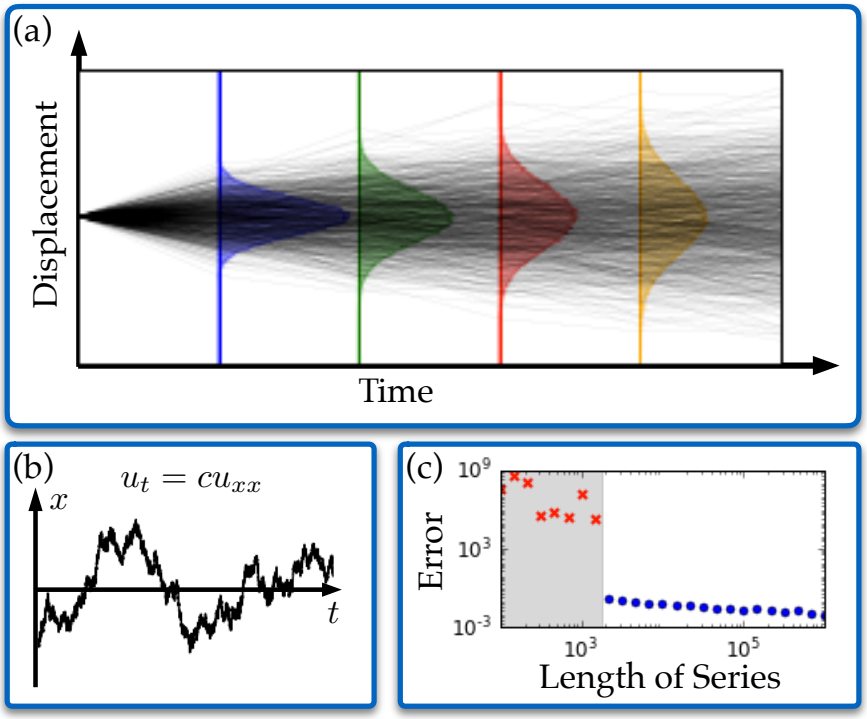


Figure 2.9: Inferring the diffusion equation from a single Brownian motion. (**a**) Time series is broken into many short random walks that are used to construct histograms of the displacement. (**b**) A single stochastic realization of Brownian motion. (**c**) Parameter error ($\|\xi^* - \hat{\xi}\|_1$) vs. length of known time series. Blue symbols correspond to correct identification of the structure of the diffusion model, $u_t = cu_{xx}$.

Since the approximate distribution function created via histograms was inherently noisy, we used polynomial interpolation to differentiate in the spatial dimension. However, since there were very few timesteps, we were unable to differentiate accurately with respect to time using polynomial interpolation and instead used finite differences for the time derivatives.

Figure 2.9(c) shows the average ℓ^1 parameter error ($\|\hat{\xi} - \xi^*\|_1$) in identifying the diffusion equation across 10 trials for various lengths of time series. For very short time series PDE-FIND had a tendency to incorrectly identify the sparsity pattern of the PDE. In many of these cases, the coefficients in the right hand side were very large. For example, in one

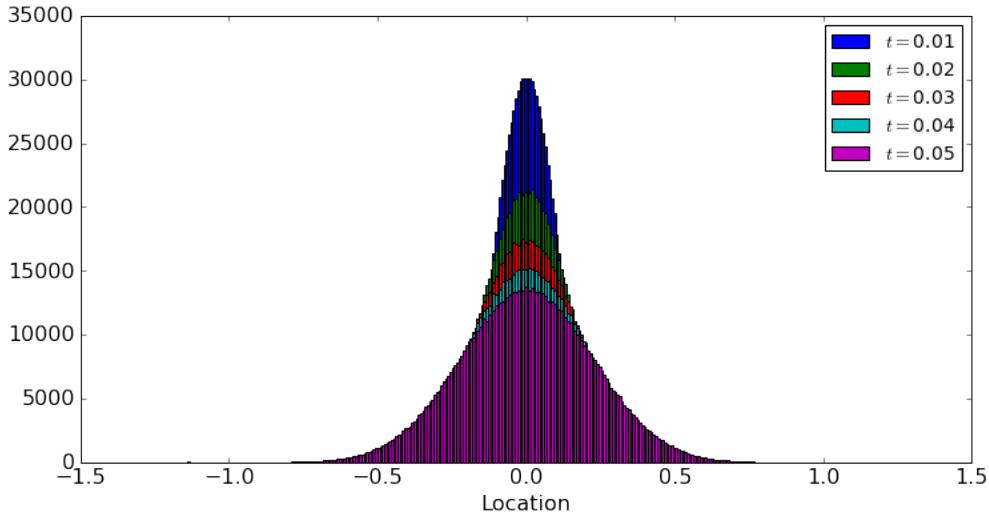


Figure 2.10: Five empirical distributions are presented illustrating the statistical spread of a particle's expected location over time. The data used to construct these histograms is collected from a single Brownian motion trajectory, see Figure 2.9b.

misidentification using a time series of length 681, the identified PDE was,

$$u_t = -2776u^2 + 753361u^2u_x + 44531u^2u_{xx}.$$

For shorter time series in which the sparsity was correctly identified, the error was often from PDE-FIND underestimating the value of the coefficient for diffusion. This results in an $\mathcal{O}(1)$ error. As a result the parameter error averaged over many trials was either very large (when PDE was misidentified) or no larger than $\mathcal{O}(1)$. This is illustrated in Figure 2.9(c) where we see unpredictable error to a point before the PDE is consistently identified, then decreasing error as PDE-FIND is better able to identify the diffusion coefficient. For longer time series, coefficient error was closer to $\mathcal{O}(10^{-3})$.

PDE-FIND was also used to identify the Fokker-Planck equation for Brownian motion with a bias. That is $x(t+dt) \sim \mathcal{N}(x(t)+cdt, dt))$ A time series was generated by adding Gaussian random variables at each timestep along with a drift term cdt with $c = 2$. This

is the same as taking,

$$x_{n+1} \sim \mathcal{N}(x_n + cdt, dt), \quad c = 2.$$

Histograms used for approximating the distribution function are shown in Figure 2.9a. The Fokker-Planck equation for the distribution function is,

$$u_t + cu_x = 0.5u_{xx}.$$

Normally, the best results in the PDE-FIND algorithm were obtained using sequential threshold ridge regression with columns of Θ normalized to have unit variance, however, that approach was the worst in the case of identifying the advection diffusion equation from a biased random walk, failing to identify the advection term. Sequential threshold ridge without normalization, LASSO, and the forward-backward greedy algorithm all correctly identified the PDE.

$u_t = 0.500872u_{xx}$	STRidge with normalization
$u_t = -2.000641u_x + 0.503582u_{xx}$	STRidge without normalization, LASSO, and Greedy

These examples, of course, are trivial and may be easily derived using the property that increments are Gaussian. However, they highlight the ability of the method to study density functions formed by binning short trajectories. In particular, the PDE-FIND framework may be suitable for studying the time evolving spatial density of interacting particles such as bacteria.

2.2.3 Limitations

PDE-FIND with an incomplete library

In applying the PDE-FIND algorithm we assume that the column space of Θ is sufficiently rich to have a sparse representation of the time dynamics of the dataset. However, when

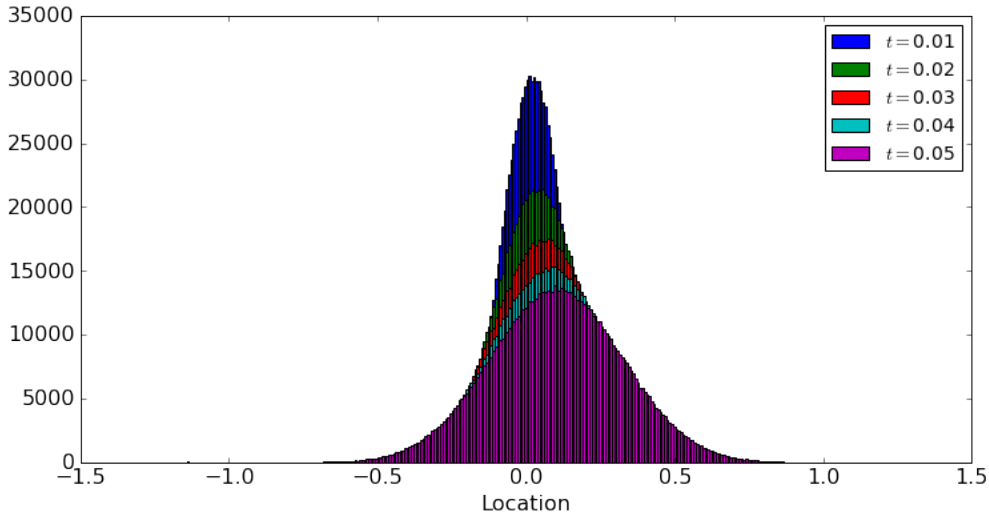


Figure 2.11: Five empirical distributions are presented illustrating the statistical spread of a particle's expected location over time with constant drift.

applying the algorithm to a data set where the dynamics are in fact unknown it is not unlikely that the column space of Θ is insufficient. We cannot provide a comprehensive analysis of what the PDE-FIND algorithm will converge to but will provide two examples of where this is the case.

Kuramoto-Sivashinsky equation with incomplete library

The Kuramoto-Sivashinsky equation involves a fourth spatial derivative, which one may not guess to allow when looking for models to fit the observed dynamics. We tried identifying an equation using the same numerical solution to the Kuramoto-Sivashinsky equation but not allowing for fourth order derivatives. The result was that the u_{xxxx} was replaced with a uu_{xxx} term. Each of the other two terms was correctly identified but with large coefficient error.

$$u_t = -0.189uu_x + 0.108u_{xx} - 0.076uu_{xxx} \tag{2.14}$$

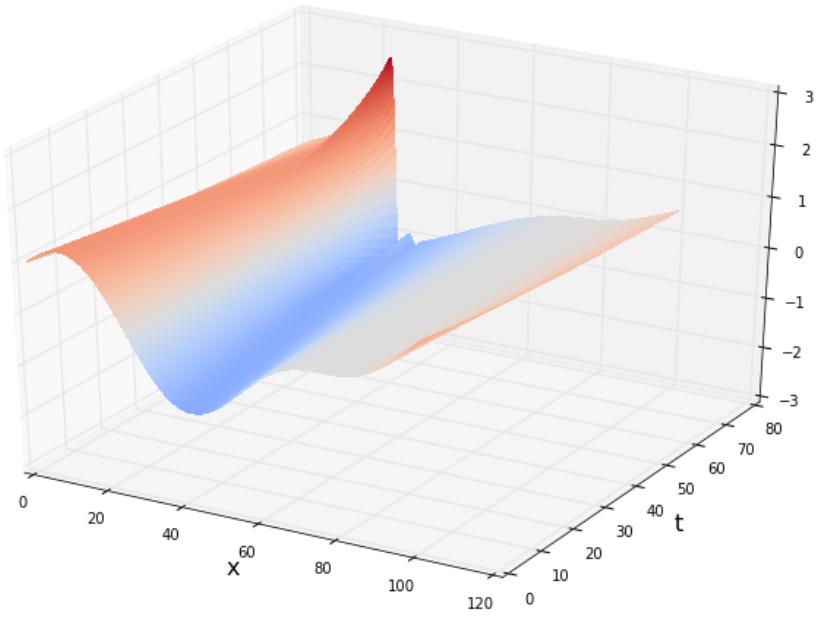


Figure 2.12: The numerical solution to the misidentified Kuramoto-Sivashinsky equation given by (2.14).

While uu_{xxx} was positively correlated with u_{xxxx} on the training data the dynamics are radically different. A numerical solution to the misidentified PDE is shown in figure 2.12.

Nonlinear Schrödinger equation with incomplete library

The Nonlinear Schrödinger equation relates the temporal derivative of u not only to u and its spatial derivatives but also to $|u|$. We tried fitting the solution to the NLS equation to a library of candidate functions that does not contain any power of $|u|$. Unlike the Kuramoto-Sivashinsky example, the result did not appear related to the true equation. The resulting PDE is,

$$\begin{aligned} u_t =&(0.368545 - 0.386346i) + (-0.000381 + 0.100149i)u_{xx} + (-0.233212 - 0.242627i)u^2 \\ &+ (0.066812 - 0.000737i)u^3 + (-0.001855 + 3.796550i)u + (0.024974 - 0.000055i)u^2u_{xx} \\ &+ (0.001612 - 0.001612i)u^3u_{xx} + (0.059966 + 0.059281i)uu_{xx} \end{aligned} \tag{2.15}$$

A numerical solution obtained via spectral differentiation and ODE-45 to the misidentified PDE for the NLS data is shown in figure 2.13.

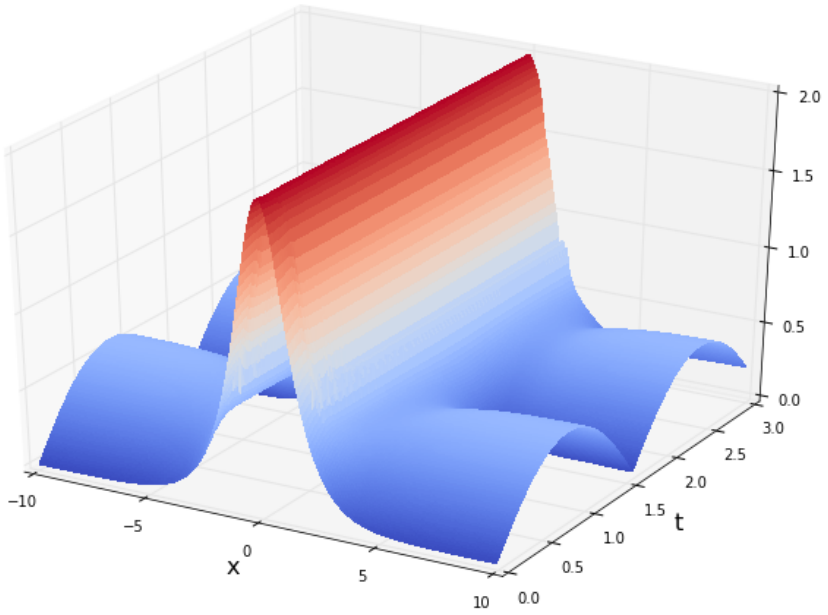


Figure 2.13: The numerical solution to the misidentified Nonlinear Schrödinger equation given by (2.15).

Higher levels of noise

Identifying governing equations with noisy datasets is made difficult due to the challenge in numerically differentiating noisy data. We have provided examples of this for each of

the canonical PDEs discussed in this thesis but did not exceed 1% noise. Figure 2.14 shows the error in identifying Burgers' equation with noise levels of up to 12%. In this case, PDE-FIND is able to correctly identify the correct terms in the PDE with modestly increasing error until the noise level reaches 10%, at which point we see extra terms added to the identified model. At 12%, every candidate function is included in the identified dynamics.

	Candidate Functions															
Noise Level	1	u	u^2	u^3	u_x	uu_x	u^2u_x	u^3u_x	u_{xx}	uu_{xx}	u^2u_{xx}	u^3u_{xx}	u_{xxx}	uu_{xxx}	u^2u_{xxx}	u^3u_{xxx}
Clean						−0.986			0.119							
1%						−0.986			0.119							
2%						−0.986			0.118							
3%						−0.986			0.118							
4%						−0.985			0.118							
5%						−0.984			0.117							
6%						−0.983			0.117							
7%						−0.982			0.116							
8%						−0.980			0.115							
9%						−0.978			0.114							
10%				0.168		−1.221			0.076	0.237					−0.090	
11%						−1.127			0.086	0.115					−0.073	
12%	0.001	0.054	−0.239	0.319	−0.064	−0.950	0.190	−0.449	0.035	0.591	−1.179	0.942	−0.006	0.081	−0.300	0.178

Figure 2.14: Results of PDE-FIND applied to Burgers' equation for varying levels of noise.

The reaction diffusion equation was much less robust to noise. Even at 1%, PDE-FIND misidentifies the terms in the PDE. The true model, as well as identified models with clean data, 0.5% noise, and 1% noise are given below. Note that coefficients for clean data are inexact but have been rounded to three decimal points.

$$\left.\begin{aligned} u_t &= 0.1u_{xx} + 0.1u_{yy} - uv^2 - u^3 + v^3 + u^2v + u \\ v_t &= 0.1v_{xx} + 0.1v_{yy} - uv^2 - u^3 - v^3 - u^2v + v \end{aligned}\right\} \text{True model}$$

$$\left.\begin{aligned} u_t &= 0.100u_{xx} + 0.100u_{yy} - 1.000uv^2 - 1.000u^3 + 1.000v^3 + 1.000u^2v + 1.000u \\ v_t &= 0.100v_{xx} + 0.100v_{yy} - 1.000uv^2 - 1.000u^3 - 1.000v^3 - 1.000u^2v + 1.000v \end{aligned}\right\} \text{Clean Data}$$

$$\left.\begin{aligned} u_t &= 0.095u_{xx} + 0.095u_{yy} - 0.945uv^2 - 0.945u^3 + 1.000v^3 + 1.000u^2v + 0.945u \\ v_t &= 0.095v_{xx} + 0.095v_{yy} - 1.000uv^2 - 1.000u^3 - 0.946v^3 - 0.946u^2v + 0.946v \end{aligned}\right\} \text{0.5\% Noise}$$

$$\left.\begin{aligned} u_t &= 0.077u_{xx} + 0.077u_{yy} - 0.731uv^2 - 0.732u^3 + 0.810v^3 + 0.810u^2v + 0.776u + 0.169v \\ v_t &= 0.081v_{xx} + 0.079v_{yy} - 0.832uv^2 - 0.832u^3 - 0.772v^3 - 0.772u^2v + 0.815v - 0.149u \end{aligned}\right\} \text{1\% Noise}$$

Limited Data

In each of the examples provided, we have far more data points than library functions. The importance of using a large number of points lies more in the numerical evaluation of derivatives than in supplying sufficient data for the regression. In table 2.9 we test the PDE-FIND method on a number of discretizations of a solution to Burgers' equation. An initial solution was computed on a fine grid and data points from the fine solution were taken on courser grids to evaluate the performance of the method with courser sampling. As expected, we notice a decline in accuracy with courser grids and eventually the inability to accurately identify the true sparsity pattern of the coefficient vector. We should note though that this is not due to a lack of points in the regression. Using the original grid (1024 x 512) to evaluate derivatives we were able accurately identify the PDE using just 10 points with 0.023% error in the coefficients.

2.2.4 Non-uniform data

The PDE-FIND algorithm requires numerical approximations of the spatial and temporal derivatives of data in order to construct a sparse regression problem for the derivatives.

Spatial Points: n	Temporal Points: m				
	256	128	64	32	16
512	0.113	0.153	0.896	2.446	
256	0.777	0.509	0.238		
128	3.417	3.140		1.161	
64	13.72				
32					

Table 2.9: Accuracy of PDE-FIND on Burger's equation with various grid sizes. Red table entries denote a misidentification of the sparisty pattern either due to the inclusion of extra terms or missing one of the two terms in Burgers' equation. Blue entries show average parameter error as percent of true value. Measurements on all grids were taken from numerical solution on fine grid to ensure error in the method is intrinsic to PDE-FIND and not the numerical solution of Burgers' equation.

Traditional methods for numerical differentiation such as finite differences, Fourier transforms, and polynomial interpolation generally require regularly spaced data. In practice, sensor placement for a real system may not allow for this regularity. While not explored in this work, we highlight one particular method used for interpolating irregularly spaced data that may be useful for the practical implementation of PDE-FIND. The Physics Informed Neural Network paradigm, introduced by Raissi *et al.* [132, 133] uses neural networks to interpolate data, allowing for automatic differentiation of the neural network to compute derivatives.

2.3 Parametric Equations

In this section, we present a sparse regression framework for identifying PDEs with non-constant coefficients. Specifically, we allow for variation in the value of a coefficient across time or space, but maintain that the active terms in the PDE must be consistent. This is an important innovation in practice as the parameters of physical systems often vary during the measurement process, so that the parametric dependencies be disambiguated from the PDE itself. This method extends the sparse regression framework developed in Sec. 2.2 and references [139, 144] by using group sparsity. We are still limited by the accuracy

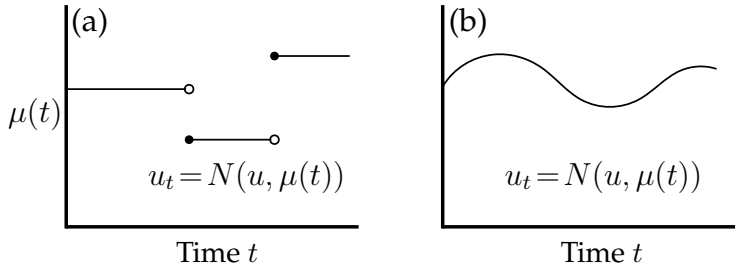


Figure 2.15: Two prototypical scenarios for parametric model discovery: (a) Parameters $\mu(t)$ are piece wise constant and change at fixed points in time, (b) The underlying PDE model $u_t = N(u, \mu(t))$ depends on continuously varying parameter $\mu(t)$.

of numerical differentiation and by the library terms in the sparse regression. Numerical differentiation using neural networks as shown in [129] appears promising as a method for obtaining more accurate time derivatives from noisy data. The limitation based on terms included in the library seems more permanent. Any closed-form model expression must be representable with a finite set of building blocks. Here we only use monomials in our data and its derivatives, since these are the common terms seen in physics, but there is no limitation to the terms included in the library.

Figure 2.15 demonstrates two prototypical parametric dependencies: (a) a PDE model whose constant parameters change at fixed points in time and (b) a PDE model that depends continuously on the parameter $\mu(t)$. Our proposed model discovery method provides a principled approach to efficiently handle these two parametric cases, thus advancing the field of model discovery by disambiguating the dynamics from parametric time dependence. Even if the governing equations are known, parametric dependencies in PDEs complicate numerical simulation schemes and challenge one's ability to produce accurate future state predictions. Characterizing parametric dependence is also an critical task for model reduction in both time-independent [127, 57] and time-dependent PDEs [7, 8]. Thus the ability to explicitly extract the parametric dependence of a spatio-temporal system is necessary for accurate, quantitative characterization of PDEs.

Group Sparsity

In a typical sparse regression, we seek a sparse solution to the linear system of equations $\mathbf{Ax} = \mathbf{b}$. Accuracy of the predictor, $\|\mathbf{Ax} - \mathbf{b}\|$, is balanced against the number of nonzero coefficients in $\mathbf{x}$. Thus the sparse regularization enforces a solution $\mathbf{x}$ with many zeros (which is the variable ξ in (2.3a)). In this work, we use the notion of group sparsity to find time series representing each parameter in the PDE, rather than single values. We group collections of terms in $\mathbf{x}$ together and seek solutions to $\mathbf{Ax} = \mathbf{b}$ that minimize the number of groups with nonzero coefficients.

One well studied method for solving regression problems with group sparsity is the group LASSO [46], given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{w}} \frac{1}{2n} \left\| \mathbf{b} - \sum_{g \in \mathcal{G}} \mathbf{A}_{(g)} \mathbf{w}_{(g)} \right\|_2^2 + \lambda \sum_{g \in \mathcal{G}} \|\mathbf{w}_{(g)}\|_2. \tag{2.16}$$

Here $\mathcal{G}$ is a collection of groups, each of which contains a subset of the indices enumerating the columns of $\mathbf{A}$ and coefficients in $\mathbf{x}$. Note that the second term in the group LASSO corresponds to a convex relaxation of the number of groups containing a nonzero value, given by $\sum_{g \in \mathcal{G}} \delta_{\|\mathbf{w}_{(g)}\|,0}$. Optimal solutions to (4) are characterized by the Karush-Kuhn-Tucker (KKT) conditions. For each $g' \in \mathcal{G}$, solutions must satisfy

$$\underbrace{\frac{1}{n} {\mathbf{A}_{(g')}}^T \left(\mathbf{b} - \sum_{g \neq g'} \mathbf{A}_{(g)} \mathbf{x}_{(g)} \right)}_{\mathbf{r}_{(g')}} - \frac{1}{n} {\mathbf{A}_{(g')}}^T \mathbf{A}_{(g')} \mathbf{x}_{(g')} \in \lambda \partial \|\mathbf{x}_{(g')}\|_2, \tag{2.17}$$

where the subgradient is a set valued function $\partial \|\mathbf{x}_{(g')}\|_2 = \{\mathbf{x}_{(g')} \|\mathbf{x}_{(g')}\|_2^{-1}\}$ if $\mathbf{x}_{(g')} \neq 0$ or $\partial \|\mathbf{x}_{(g')}\|_2 = \{\mathbf{y} : \|\mathbf{y}\|_2 \leq 1\}$ for $\mathbf{x}_{(g')} = 0$. The KKT condition reveals that there is a solution with $\mathbf{x}_{(g')} = 0$ when $\|\mathbf{r}_{(g')}\|_2 \leq \lambda$. Conditions for uniqueness of the solution to (2.16) are explored in [135].

The concept of group sparsity has been used in several previous methods for identi-

fying dynamics given by ordinary differential equations [26, 147]. As it will be shown in the following, we find that the group LASSO performs poorly in the case of identifying PDEs. We instead use a sequential thresholding method based on ridge regression, similar to the method used in [139], but adapted for group sparsity. A sequential thresholding method was also used in [147] for group sparsity but for ordinary and not partial differential equations. [147] also provides some theoretical results concerning the convergence of a hard thresholding algorithm for group sparsity while the case of hard thresholding for non-group sparsity is extensively studied in [181]. Our method, which we call Sequential Grouped Threshold Ridge Regression (SGTR), is summarized in algorithm 3.

Algorithm 3 SGTR($\mathbf{A}, \mathbf{b}, \mathcal{G}, \lambda, \epsilon,$ maxit, $f(\mathbf{x}) = \|\mathbf{x}\|_2$)

Solves $\mathbf{x} \approx \mathbf{A}^{-1}\mathbf{b}$ with sparsity imposed on groups in $\mathcal{G}$

Initialize coefficients with ridge regression

$\mathbf{x} = \arg\min_{\mathbf{w}} \|\mathbf{b} - \mathbf{A}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$

Threshold groups with small f and repeat

for $iter = 1, \ldots,$ maxit:

    # Remove groups with sufficiently small $f(\mathbf{x}_{(g)})$

    $\mathcal{G} = \{g \in \mathcal{G} : f(\mathbf{x}_{(g)}) > \epsilon\}$

    # Refit these groups (note this sets $\mathbf{x}_{(g)} = 0$ for $g \notin \mathcal{G}$)

    $\mathbf{x} = \arg\min_{\mathbf{w}} \|\mathbf{b} - \sum_{g\in\mathcal{G}} \mathbf{A}_{(g)}\mathbf{w}_{(g)}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$

Get unbiased estimates of coefficients after finding sparsity

$\mathbf{x}_{(\mathcal{G})} = \arg\min_{\mathbf{w}} \|\mathbf{b} - \sum_{g\in\mathcal{G}} \mathbf{A}_{(g)}\mathbf{w}_{(g)}\|_2^2$

return $\mathbf{x}$

Throughout the training, $\mathcal{G}$ tracks the groups that have nonzero coefficients, and it is paired down as we threshold coefficients with sufficiently small relevance, as measured by f. If a group $g \in \mathcal{G}$ meets the thresholding criterion $f(\mathbf{x}_{(g)}) < \epsilon$ then $\mathbf{x}_{(g)}$ is set to zero and not considered in subsequent iterations of algorithm 1. We use the 2-norm of

the coefficients in each group for f but one could also consider arbitrary functions. In particular, for problems where the coefficients within each group have a natural ordering, as they do in our case as time series or spatial functions, one could consider smoothness or other properties of the functions. In practice, we normalize each column of $\mathbf{A}$ and $\mathbf{b}$ so that differences in scale between the groups do not affect the result of the algorithm. For the group LASSO we always perform an unregularized least squares regression on the nonzero coefficients after the sparsity pattern has been discovered to de-bias the coefficients. We found SGTR to outperform the group LASSO for the problem of correctly identifying the active terms in parametric PDEs.

Data-driven identification of parametric partial differential equations

In the identification of parametric PDEs, we consider equations of the form

$$u_t = N(u, u_x, \ldots, \mu(t)) = \sum_{j=1}^{d} N_j(u, u_x, \ldots)\xi_j(t). \tag{2.18}$$

Note that this equation is similar to (2.4) but has time-varying parametric dependence. To capture spatial variation in the coefficients, we simply replace $\xi(t)$ with $\xi(x)$. The PDE is assumed to contain a small number of active terms, each with a time varying coefficient $\xi(t)$. We seek to solve two problems: (i) determine which coefficients are nonzero and (ii) find the values of the coefficients for each ξ_j at each timestep or spatial location for which we have data.

Prior knowledge of an appropriate set of basis functions may be vital to the accuracy of the algorithm proposed in this work. In what follows we assume the true dynamics N to lie in the span of our candidate functions $\{N_j\}_{j=1}^{d}$. It is shown in [18] that ODEs with trigonometric functions may be approximated by a sparse regression to their Taylor series, though no such analysis has been done for PDEs. The supplementary material in [139] demonstrates that regression using an incomplete library yields unpredictable results.

Indeed, [18] indicates that if the correct model term(s) is not present, sparse identification often fails in a stereotypical way, producing a non-parsimonious representation of the dynamical system. Thus it typically easy to determine if the algorithm fails.

For time dependent problems, we construct a separate regression for each timestep, allowing for variation in the PDE between timesteps. Similar to the PDE-FIND method, we construct a library of candidate functions for the PDE using monomials in our data and its derivatives so that

$$\boldsymbol{\Theta}\left(u^{(j)}\right) = \begin{pmatrix} | & | & | & | \\ 1 & u^{(j)} & \dots & u^3 u_{xxx}^{(j)} \\ | & | & | & | \end{pmatrix} \tag{2.19}$$

where the set of m equations is given by

$$u_t^{(j)} = \boldsymbol{\Theta}\left(u^{(j)}\right)\xi^{(j)},\ j = 1, \dots, m\,. \tag{2.20}$$

Our goal is to solve the set of equations given by (2.20) with the constraint that each $\xi^{(j)}$ is sparse and that they all share the same sparsity pattern. That is, we want a fixed set of active terms in the PDE. To do this, we consider the set of equations as a single linear system and use group sparsity. Expressing the system of equations for the parametric equation as a single linear system we get the block diagonal structure given by

$$\underbrace{\begin{pmatrix} u_t^{(1)} \\ u_t^{(2)} \\ \vdots \\ u_t^{(m)} \end{pmatrix}}_{u_t} = \underbrace{\begin{pmatrix} \boldsymbol{\Theta}\left(u^{(1)}\right) & & & \\ & \boldsymbol{\Theta}\left(u^{(2)}\right) & & \\ & & \ddots & \\ & & & \boldsymbol{\Theta}\left(u^{(m)}\right) \end{pmatrix}}_{\boldsymbol{\Theta}} \underbrace{\begin{pmatrix} \xi^{(1)} \\ \xi^{(2)} \\ \vdots \\ \xi^{(m)} \end{pmatrix}}_{\xi}. \tag{2.21}$$

We solve (2.21) using SGTR with columns of the block diagonal library matrix grouped by their corresponding term in the PDE. Thus for m timesteps and d candidate functions

in the library, groups are defined as $\mathcal{G} = \{\{j + d \cdot i : i = 0, \ldots, m-1\} : j = 1, \ldots, d\}$. This ensures a sparse solution to the PDE while also allowing arbitrary time series for each variable. For problems with spatial, rather than temporal, variation in the coefficients, we simply group by spatial rather than time coordinate. A similar block diagonal structure is obtained but with n blocks of size $m \times d$ rather than m blocks of size $n \times d$. The groups are defined by $\mathcal{G} = \{\{j + d \cdot i : i = 0, \ldots, n-1\} : j = 1, \ldots, d\}$.

Since we are evaluating the relevance of groups based on their norm, it is important to consider differences in the scale of the candidate functions. For example, if $u \sim \mathcal{O}(10^{-2})$ then a cubic function will be $\mathcal{O}(10^{-6})$ and relatively large coefficients multiplying this data may not have a large effect on the dynamics, but will not be removed by the hard threshold due to its size. To remedy this, we normalize each candidate function represented in $\boldsymbol{\Theta}$ as well as each $u_t^{(j)}$ to have unit length prior to the group thresholding algorithm and then correct for the normalization after we have discovered the correct sparsity pattern.

Model Selection

For each model we test both the group LASSO as well as SGTR using an exhaustive range of parameter values. Let $\tilde{\boldsymbol{\Theta}}$ denote the block diagonal matrix $\boldsymbol{\Theta}$ shown in equation (2.21) but with all columns normalized to have unit length and $\tilde{u}_t$ be the vector of all time derivatives having been normalized to unit length so that $\|\tilde{u}_t\| = \sqrt{m}$. For the group LASSO, we find the minimal value of λ that will set all coefficients to zero which is given by

$$\lambda_{\text{max}} = \max_{g \in \mathcal{G}} \frac{1}{n} \|\tilde{\boldsymbol{\Theta}}_{(g)}^T \tilde{u}_t\|_2. \tag{2.22}$$

We check 50 evenly spaced values of λ between $10^{-5}\lambda_{\text{max}}$ and λ_{max} on a logarithmic scale.

For SGTR, we search over the range of tolerances between ϵ_{min} and ϵ_{max} defined as

$$\epsilon_{max/min} = \underset{g \in \mathcal{G}}{max/min} \, \|\xi_{\text{ridge},(g)}\|_2, \tag{2.23}$$

where $\xi_{\text{ridge}} = (\tilde{\mathbf{\Theta}}^T\tilde{\mathbf{\Theta}} + \lambda I)^{-1}\tilde{\mathbf{\Theta}}^T\tilde{u}_t$. Note that by definition, ϵ_{min} is the minimum tolerance that has any effect on the sparsity of the predictor and ϵ_{max} is the minimum tolerance that guarantees all coefficients to be zero. A set of 50 intermediate tolerances, equally spaced on a logarithmic scale, is tested between ϵ_{min} and ϵ_{max}. Note that this technique circumvents some of the earlier assumptions regarding the thresholding algorithm implicit in algorithm 2.

To select the optimal model generated via each method, we evaluate the models using the Akaike Information Criterion (AIC) [3] inspired loss function

$$\mathcal{L}(\xi) = N \ln\left(\frac{\|\tilde{\mathbf{\Theta}}\xi - \tilde{u}_t\|_2^2}{N} + \epsilon\right) + 2k \tag{2.24}$$

where k is the number of nonzero coefficients in the identified PDE given by the number of non-zero-elements of ξ divided by the number of time-steps, $\|\xi\|_0/m$, and N is the number of rows in $\mathbf{\Theta}$, which is equal to the size of our original dataset u.

Equation (2.24) is closely related to the AIC. Typically, the mean square error of a linear model is used to evaluate goodness of fit, but in our case there is error in computing the time derivative u_t, so we assume that any linear model which perfectly fits the data is overfit. We have added $\epsilon = 10^{-5}$ to the mean square error of each model as a floor in order to avoid overfitting. This may also be interpreted as a ceiling for the likelihood of any given model. Without this addition, our algorithm selects insufficiently parsimonious representations of the dynamics. Explanations for the failure of AIC to select the correct model likely stem from the fact that it is meant as a means to analyze likelihood. The residual sum of squares error here should only be considered likelihood if numerical approximations of the temporal derivative are exact except corruptions by Gaussian noise, which is not the case. Hence, we apply AIC with the understanding that it is not a rigorous model selection technique, but does work well empirically with the added correction.

Figure 2.16 illustrates the loss function from equation (2.24) evaluated on models de-

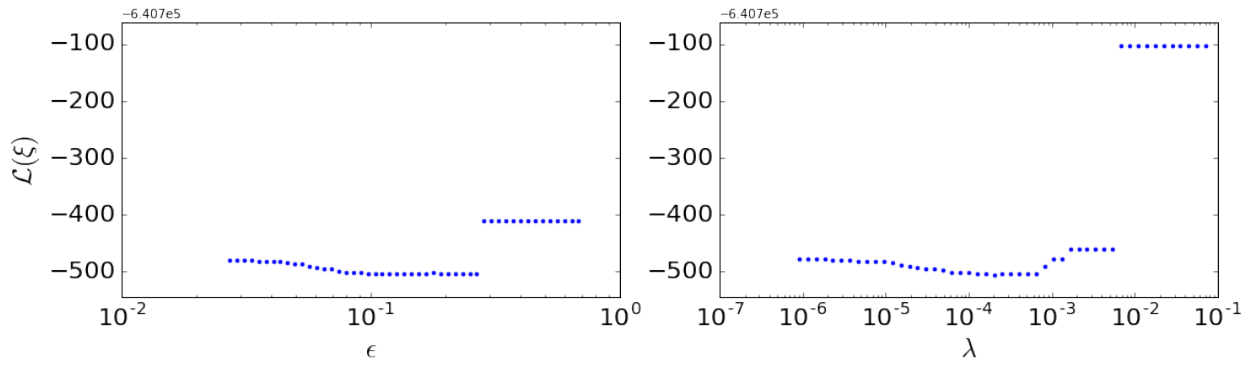


Figure 2.16: Example of loss function evaluated for a number of candidate models for the parametric Burgers' equation. Library included derivatives up to 4th order multiplying powers of u up to cubic. Left: 50 models obtained via SGTR algorithm using values of ϵ between ϵ_{min} and ϵ_{max}. Right: 50 models obtained via group LASSO for λ between $10^{-5}\lambda_{max}$ and λ_{max}.

rived from 50 values of ϵ and λ using SGTR and group LASSO respectively. Initially, a low penalty in each algorithm yields a model that is overfit to the data given our sparsity criteria. For an intermediate value of ϵ or λ, a more parsimonious but still predictive model is obtained. For sufficiently high values, the model is too sparse and is no longer predictive.

2.3.1 Computational Results of Parametric PDE Discovery

We test our method for the discovery of parametric partial differential equations on four canonical models; Burgers' equation with a time varying nonlinear term, the Navier-Stokes equation for vorticity with a jump in Reynolds number, a spatially dependent advection equation, and a spatially dependent Kuramoto-Sivashinsky equation. In each case the method is also tested after introducing white noise with mean magnitude equal to 1% of the ℓ^2-norm of the dataset. Noise is added directly to the data, $\mathbf{U}$ prior to numerical differentiation in order to replicate the effects of sensor noise. The proposed method is able to accurately identify the dynamics in each case except the Kuramoto-Sivashinsky

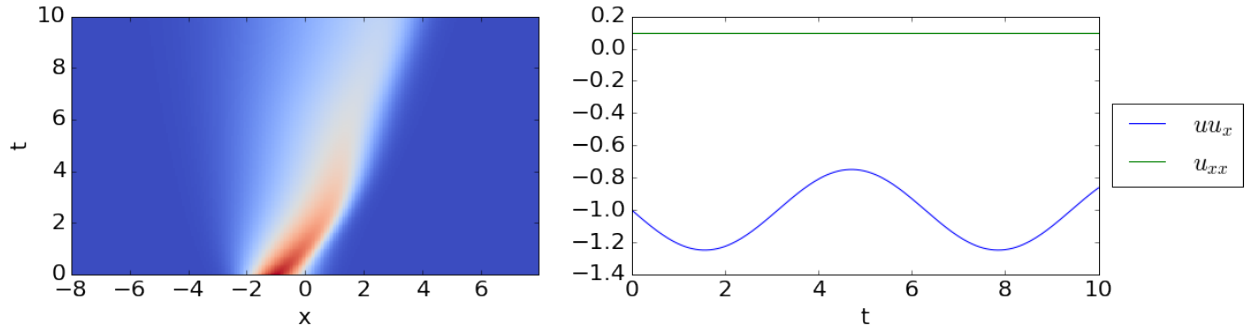


Figure 2.17: Left: dataset for identification of the parametric diffusive Burgers' equation. Here the PDE was evolved on the interval $[-8, 8]$ with periodic boundary conditions and $t \in [0, 10]$. Right: Coefficients for the terms in the parametric Burgers' equation. The diffusion was held constant at 0.1 while the nonlinear advection as coefficient is given by $a(t) = -(1 + sin(t)/4)$.

equation, where the inclusion of a fourth order derivative makes numerical evaluation with noise highly challenging. Since final coefficient values for both group LASSO and SGTR are found using an unregularized regression on the non-zero terms found using sparse regression, results in any case where both algorithms determine the correct active terms are identical. We have therefore omitted side by side comparison and instead only show the group LASSO coefficients in cases where that method failed to identify the correct active terms.

Burgers' Equation

To test the parametric discovery of PDEs, we consider a solution of Burgers' equation with a sinusoidally oscillating coefficient $a(t)$ for the nonlinear advection term

$$\begin{aligned} u_t &= a(t)uu_x + 0.1u_{xx} \\ a(t) &= -\left(1 + \frac{\sin(t)}{4}\right) \end{aligned} \tag{2.25}$$

where a small amount of diffusion is added to regularize the evolution dynamics.

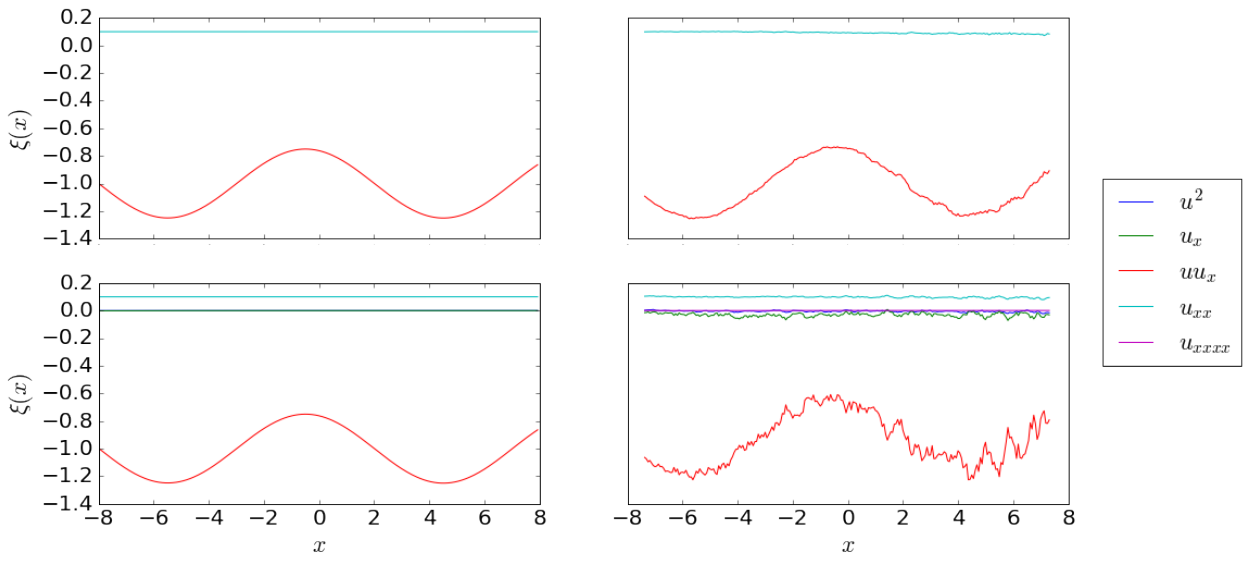


Figure 2.18: Time series discovered for the coefficients of the parametric Burgers' equation. Top row: SGTR method, which correctly identifies the two terms. Bottom row: group LASSO method which adds several additional (incorrect) terms to the model. The left panels are noise-free, while the right panels contain 1% noise.

Burgers' equation is solved numerically using the discrete Fourier transform (DFT) to evaluate spatial derivatives and the SciPy function *odeint* [70] for temporal integration on the interval $[-8, 8]$ with periodic boundary conditions and $t \in [0, 10]$ with $n = 256$ grid points and $m = 256$ time steps. We search for parsimonious representations of the dynamics by including powers of u up to cubic order, which can be multiplied by derivatives of u up to fourth order. For the noise-free dataset we use the DFT for computing derivatives. For the noisy dataset, we use polynomial interpolation to smooth the derivatives [139].

The resulting time series for the identified nonzero coefficients are shown in Fig. 2.18. SGTR correctly identified the active terms in the PDE for both the noise-free and noisy datasets, whereas group LASSO fails in both cases to produce the correct PDE and its parametric dependencies. For group LASSO, the λ minimizing (12) is not the minimum value of $\lambda_{min} = 10^{-5}\lambda_{max}$, indicating that an appropriate range has been tested. Furthermore, no value of λ yields the correct sparsity pattern. This strongly suggests that there

does not exist a λ such that the group LASSO realizes the correct sparsity pattern for the Burgers' equation dataset considered in this work. This may be due to the convex relaxation from $\delta_{\|\mathbf{w}_{(g)}\|,0}$ to $\|\mathbf{w}_{(g)}\|_2$ used in (4). We see from (5) that $\mathbf{x}_{(g')} = 0$ if $\mathbf{A}_{(g')}$ is sufficiently misaligned with $\mathbf{b} - \sum_{g \neq g'} \mathbf{A}_{(g)}\mathbf{x}_{(g)}$. For ${\mathbf{A}_{(g')}}^T\mathbf{b}$ large this may be less likely given small errors in the other coefficients. Indeed, since the Burgers' dataset is a traveling wave, $\langle u_x, u_t \rangle$ is large, which may explain why group LASSO fails to set the coefficient for u_x to zero.

Navier-Stokes: Flow around a cylinder

We consider the fluid flow around a circular cylinder by simulating the Navier-Stokes vorticity equation

$$\omega_t + \mathbf{u} \cdot \nabla\omega = \frac{1}{\nu(t)}\Delta\omega. \tag{2.26}$$

Data is generated using the Immersed Boundary Projection Method (IBPM) [156, 33] with $n_x = 449$ and $n_y = 199$ spatial points in x and y respectively, and 1000 timesteps with $dt = 0.02$. The Reynolds number is adjusted half way through the simulation from $\nu = 100$ initially to $\nu = 75$. This is representative of the fluid velocity exhibiting a sudden decrease midway through the data collection. Our library of candidate functions is constructed using up to second order derivatives of the vorticity and multiplied by up to quadratic functions of the data. To keep the size of the machine learning problem tractable, we subsample 1000 random spatial location from the wake of the cylinder to construct our library at every tenth timestep [139]. For the noise-free dataset, far fewer points are needed to accurately identify the dynamics. We suspect that with a more careful treatment of the numerical differentiation in the case of noisy data, such as that used in [129], the same would be true for the dataset with artificial noise, however such work is not the focus of this work. The identified time series for the Navier-Stokes equation are shown in Fig. 2.20. SGTR and group LASSO both correctly identify the active terms in the PDE.

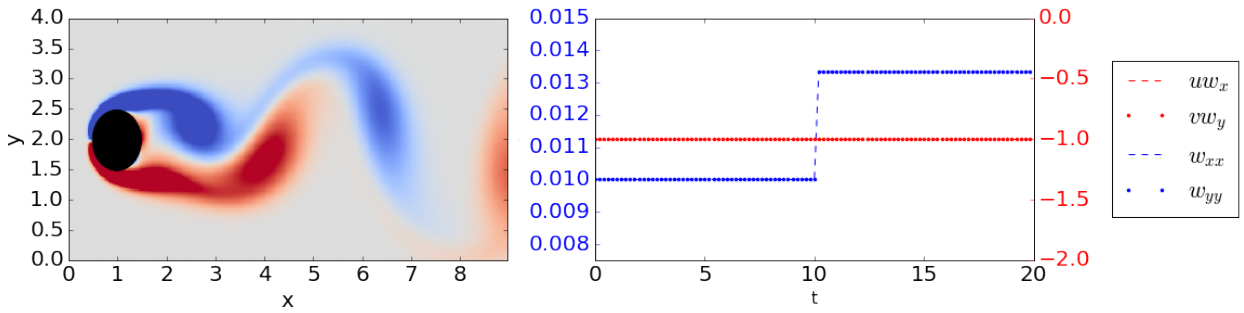


Figure 2.19: Left: dataset for identification of the parametric Navier-Stokes equation (2.26). Right: coefficients for Navier-Stokes equations exhibiting jump in Reynolds number from 100 to 75 at $t = 10$. This parametric dependency is illustrated in Fig. 2.15a.

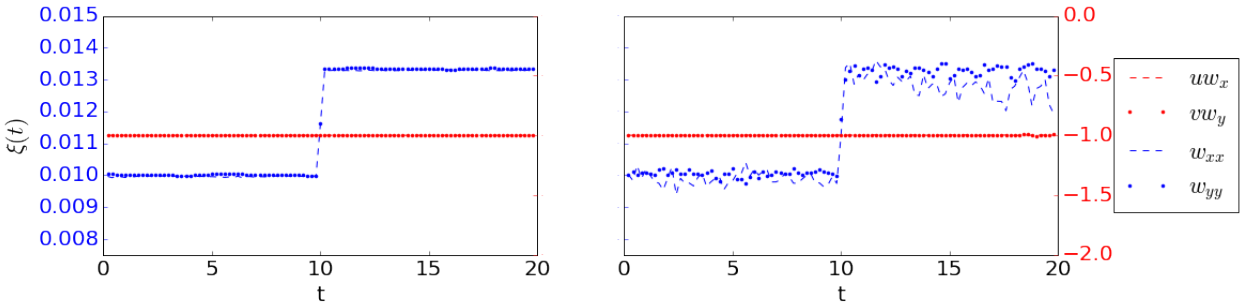


Figure 2.20: Identified time series for coefficients for the Navier-Stokes equation. Distinct axes are used to highlight jump in Reynolds number. Left: no noise. Right: 1% noise

Spatially Dependent Advection-Diffusion Equation

The advection-diffusion equation is a simple model for the transport of a physical quantity in a velocity field with diffusion. Here, we adapt the equation to have a spatially dependent velocity

$$u_t = (c(x)u)_x + \epsilon u_{xx} = c(x)u_x + c'(x)u + \epsilon u_{xx} \tag{2.27}$$

which models transport through a spatially varying vector field due to $c = c(x)$. We solve (2.27) on a periodic domain $[-L, L]$ with $L = 5$ from $t = 0$ to 5, $\epsilon = 0.1$, and $c(x) =$

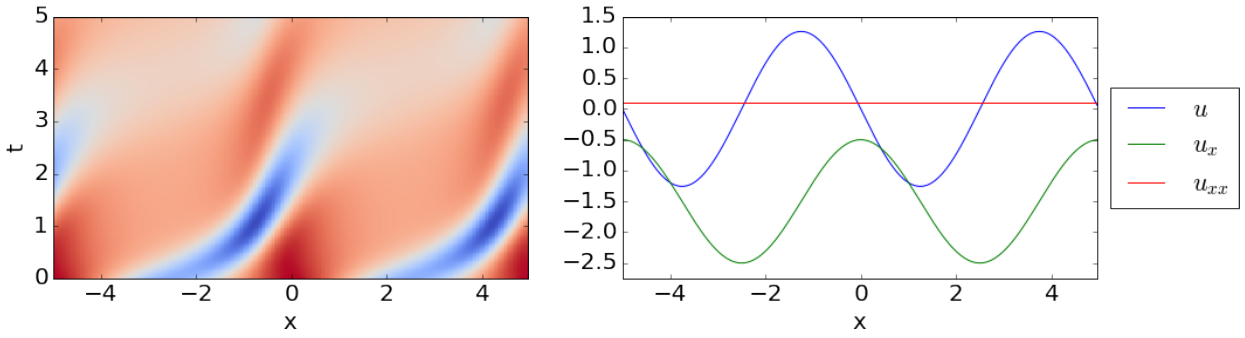


Figure 2.21: Left: dataset for identification of the spatially dependent advection diffusion equation. Right: Spatial dependence of PDE. In this case, the loadings $\xi_j(t)$ in (2.18) are replaced by $\xi_j(x)$.

$-1.5 + \cos(2\pi x/L)$ using the DFT to evaluate spatial derivatives and the SciPy function *odeint* for temporal integration with $n = 256$ and $m = 256$. The library consists of powers of u up to cubic, multiplied by derivatives of u up to fourth order.

Results for the advection-diffusion equation are shown in Fig. 2.22. In the noise-free and noisy datasets, both SGTR and group LASSO correctly identify the active terms in the PDE.

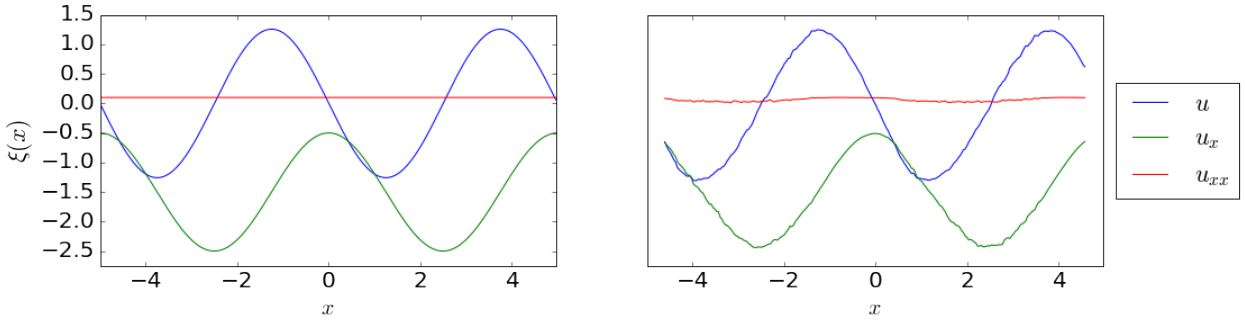


Figure 2.22: Spatial dependence of advection diffusion equation. Left: no noise. Right: 1% noise. Both SGTR and group LASSO correctly identified the active terms.

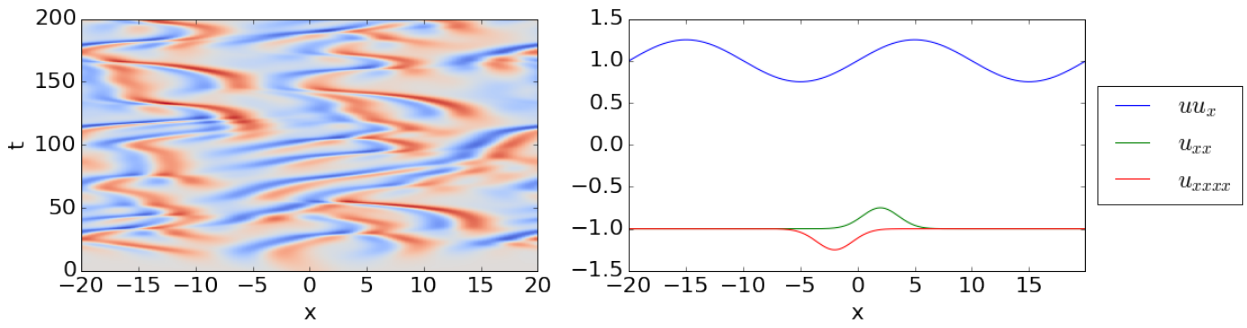


Figure 2.23: Left: dataset for identification of the spatially dependent Kuramoto-Sivashinsky equation. Right: parametric dependency of the governing equations.

Spatially Dependent Kuramoto-Sivashinsky Equation

We now test the method on a Kuramoto-Sivashinsky equation with spatially varying coefficients

$$u_t = a(x)uu_x + b(x)u_{xx} + c(x)u_{xxxx}. \tag{2.28}$$

We use a periodic domain $[-L, L]$ with $L = 20$ and coefficients $a(x) = 1 + \sin(2\pi x/L)/4$, $b(x) = -1 + e^{-(x-2)^2/5}/4$ and $c(x) = -1 - e^{-(x+2)^2/5}/4$.

We solve (2.28) numerically using the DFT to evaluate spatial derivatives and the SciPy function *odeint* for temporal integration to $t = 200$ using $n = 512$ grid points and $m = 1024$ timesteps. The second half of the data set is used, so as to only consider the region where the dynamics exhibited spatio-temporal chaos, resulting in a dataset containing 512 snapshots of 512 gridpoints. The inclusion of a fourth order derivative in the Kuramoto-Sivashinsky equation makes accurate identification difficult due to error caused by noise in numerical differentiation. Indeed, our method fails to correctly identify the active terms when 1% noise is added. With 0.01% noise the correct terms were identified but with substantial error in coefficient value. We suspect that this shortcoming could at least be partially remedied by a more careful treatment of the numerical differentiation such as in [15]. The results of our parametric identification are shown in Fig. 2.24.

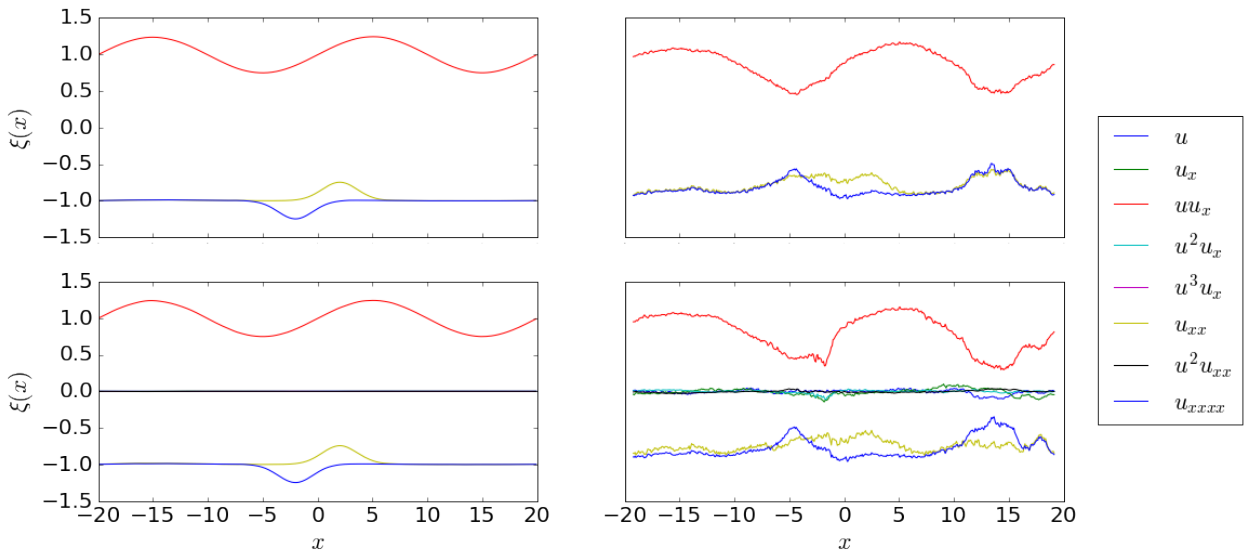


Figure 2.24: Spatial dependence of Kuramoto-Sivashinsky. top row: SGTR. Bottom row: group LASSO. Left: no noise. Right 0.01% noise using SGTR. SGTR detects correct sparsity with significant parameter error. Group LASSO does not correctly identify the parsimonious model, nor does it do a good job at predicting the correct parametric values.

2.4 Discussion

In this chapter we have presented a method for identifying governing PDEs for physical systems including those which exhibit either spatially or temporally dependent behavior. The method builds on a growing body of work in the applied mathematics and machine learning community that seeks to automate the process of discovering physical laws.

We highlight that the results in this work are dataset specific. For different geometries and initial conditions the algorithm may yield variable results and determining sufficient conditions for accurate recovery of governing equations is an open problem. For a discussion of convergence results of hard-thresholding for the constant coefficient ordinary differential equation case, we refer the reader to [181].

As is the case with other sparse regression methods for identifying dynamical systems, this method is constrained by the ability of the user to accurately differentiate data. For

ordinary differential equations, this may be circumvented by looking at the weak form of the dynamics [146], but doing so for PDEs seems difficult since there are derivatives that need to be evaluated with respect to multiple variables. We find the automatic differentiation approach used in [129] promising and suspect that the inclusion of neural network based differentiation could radically improve the ability of our method to identify dynamics from noisy data. With sufficient knowledge of data it may also be possible to obtain better estimates through tuning the polynomial based differentiation [15].

Automating the identification of closed form physical laws from data will hopefully boost scientific progress in areas where deriving the same laws from first principals proves intractable. There are several limitations to many methods proposed in the field thus far. In particular, there is a trade off between methods that are able to derive parsimonious representations, though which are limited to a finite set of library elements, and those that use black box models to represent larger classes of possible functions. Additionally, one may find difficulties in attempting to infer dynamics from the wrong set of measurements. For example, one could not derive the Schrödinger equation by only looking at measurements of intensity. While not addressing these issues, this work makes a step towards generalizing the class of equations which may be accurately identified via machine learning methods.

Chapter 3

ROBUST NEURAL NETWORK BASED FORECASTING

This chapter develops a framework for training neural networks as data-driven forecasting models in the context of noisy and limited measurements. The chapter is organized as follows: In Sec. 3.2 we outline the problem formulation, introduce our computational graph for defining a map between successive observations, describe the optimization problem used to estimate noise and dynamics, and discuss various measures of error. In Sec. 3.3 we apply our method to a selection of test problems of increasing complexity. Section 3.4 discusses pitfalls of using a black box machine learning technique such as a neural network as a model for the underlying vector field. Several examples are given to demonstrate radical differences between true and learned vector fields, even when test trajectories are accurately predicted. Section 3.5 provides a brief discussion.

3.1 Introduction

Neural networks are widely used in machine learning for tasks ranging from image recognition to language translation [54]. Their success may be attributed to the high dimensionality of the parameter space and ease of training via the back-propagation algorithm and numerical tricks for stability. More recently, advances in hardware, particularly in the use of graphics processing units for computation, and open source software such as TensorFlow [1] have lead to an explosion of interest in the use of neural networks. The renewed interest in and success of neural networks for general machine learning tasks has encouraged much greater interest in using them as tools for problems in dynamics.

The utility of neural networks as forecasting models dates back to long before they were well known in the engineering community [24]. A seminal paper by Gonzalez *et al.*

[53] laid much of the groundwork for the use of neural networks in dynamical systems by showing they could be trained within Runge-Kutta schemes to form discrete time flow maps. In this context, the neural network interpolates an unknown vector field, but is trained such that the learned vector field accurately reflects observed dynamics. More recent work has used linear multistep methods for largely the same purpose [134]. More generally, there is substantial interest in the relationship between residual networks [56], and differential equations. This differential equation viewpoint of neural networks was formalized in [173] and implemented more completely in [27]. Alternatively, recurrent neural networks naturally model sequential processes and have been used for forecasting [167, 118, 97, 121, 120, 126] and closure models for reduced order models [170, 117]. The common recurrent structure long-short term memory (LSTM) introduce memory terms that allow for effects on unresolved variables in the dynamics, but also lack the interpretation that a simple interpolation of the dynamical system's vector field has.

This chapter will develop method for training simple feed-forward networks similar to [53]. Since no memory term will be considered (as is the case with recurrent networks) it is assumed that the underlying dynamics are Markov. In contrast to previous work in neural network based forecasting models, we explicitly consider noise as a feature of the training data. By constraining the learning process inside a numerical timestepping scheme, we can improve the ability of automated methods for model discovery by cleanly separating measurement error from the underlying state while simultaneously learning a neural representation of the governing equations. A Runge-Kutta integration scheme is imposed in the optimization problem to focus the neural network to identify the continuous vector field, even when the data has a variable timestep. Our method yields predictive models on a selection of dynamical systems models of increasing complexity even with substantial levels of measurement error and limited observations. We also highlight the inherent risks in using neural networks to interpolate governing equations. In particular, we focus on overfitting and the challenge of fitting dynamics away from a bounded attractor. Trade-offs between black-box representations, such as neural networks, and

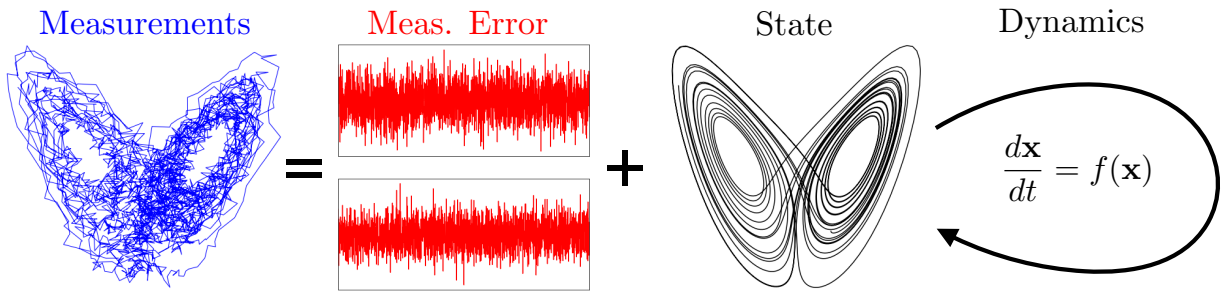


Figure 3.1: Measurements of a dynamical system may be split into measurement error and an underlying state that is governed by a dynamical system.

sparse regression methods are considered.

3.2 Methods

The methods presented in this work add to a growing body of literature concerned with the machine learning of dynamics from data. A fundamental problem in the field has been formulating methods that are robust to measurement noise. Many previous works have either relied on smoothing techniques or large datasets to average out the noise. We propose a novel approach by treating measurement error as a latent variable relating observations and a true state governed by dynamics, as in Fig. 3.1. Our method generates accurate predictive models from relatively small datasets corrupted by large amounts of noise by explicitly considering the measurement noise as part of the model instead of smoothing the data. The computational methodology simultaneously learns pointwise estimates of the measurement error, which are subtracted from the measurements to estimate the underlying state, as well as a dynamical model to propagate the true state. Section 3.2.1 provides an overview of the class of problems we consider, Sec. 3.2.2 discusses our computational approach, and Sec. 3.2.5 provides metrics to evaluate the accuracy of the data-driven model.

3.2.1 *Problem formulation*

We consider a continuous dynamical system of the form,

$$\frac{d}{dt}\mathbf{x}(t) = f\left(\mathbf{x}(t)\right), \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and f is a Lipschitz continuous vector field. This work assumes that f is unknown and that we have a set of measurements $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$ with $\mathbf{y}_j \in \mathbb{R}^n$ representing the true state at time t_j corrupted by measurement noise $\boldsymbol{\nu}_j$:

$$\mathbf{y}_j = \mathbf{x}_j + \boldsymbol{\nu}_j. \tag{3.2}$$

The discrete-time map from $\mathbf{x}_j$ to $\mathbf{x}_{j+1}$ may be written explicitly as

$$\mathbf{x}_{j+1} = F(\mathbf{x}_j) = \mathbf{x}_j + \int_{t_j}^{t_{j+1}} f(\mathbf{x}(\tau))\, d\tau. \tag{3.3}$$

Many leading methods to approximate the dynamics (e.g., f or F) from data involve a two-step procedure, where pre-processing methods are first applied to clean the data by filtering or smoothing noise, followed by the second step of fitting the dynamics. In contrast, we seek to simultaneously approximate the function f and obtain estimates of the measurement error $\boldsymbol{\nu}_j$ at each timestep. Specifically, we consider both the dynamics and the measurement noise as unknown components in a map between successive observations:

$$\mathbf{y}_{j+i} = \overbrace{F^i(\underbrace{\mathbf{y}_j - \boldsymbol{\nu}_j}_{\mathbf{x}_j})}^{\mathbf{x}_{j+i}} + \boldsymbol{\nu}_{j+i}. \tag{3.4}$$

Starting with the observation $\mathbf{y}_j$, if the measurement noise $\boldsymbol{\nu}_j$ is known, it may be subtracted to obtain the true state $\mathbf{x}_j$. The flow map is applied i times to the state $\mathbf{x}_j$ to obtain the true state at timestep $\mathbf{x}_{j+i}$, and adding the measurement noise back will yield the observation $\mathbf{y}_{j+i}$. Given that $\mathbf{y}_j$ is observed, there are two unknown quantities in

(3.4): the dynamics F and the measurement noise $\mathbf{N} = [\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m]$. We leverage the fact that (3.4) must hold for all pairs of observations including $i < 0$ and that $\mathbf{x}(t)$ is autonomous and Markovian to effectively learn both dynamics and measurement error. In this framework, consistency of governing dynamics helps decompose the dataset into state and measurement error, while explicit estimates of measurement error simultaneously allow for a more accurate model of the dynamics. In addition, we approximate F with a Runge-Kutta scheme and focus on modeling the continuous dynamics f, which is generally simpler than F.

3.2.2 Computational framework

Here we outline a method for estimating both the unknown measurement noise and model in (3.4), shown in Fig. 3.2. We let $\hat{\mathbf{N}} = [\hat{\boldsymbol{\nu}}_1, \ldots, \hat{\boldsymbol{\nu}}_m] \in \mathbb{R}^{n \times m}$ and $\hat{F}_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$ denote the data-driven approximation of the measurement noise and discrete-time dynamics, respectively. We develop a cost function to evaluate the fidelity of model predictions in comparison with the observed data and then we solve an optimization problem to refine estimates for both the measurement noise $\hat{\mathbf{N}}$ and the parameterization of $\hat{F}_{\boldsymbol{\theta}}$. The map $\hat{F}_{\boldsymbol{\theta}}$ is modeled by a Runge-Kutta time-stepper scheme that depends on a continuous vector field, f, which is modeled by a feed-forward network.

We construct a data-driven model for the vector field f using a feed-forward neural network:

$$\hat{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \left(\prod_{i=1}^{l} C_{g_i} \right) (\mathbf{x}) \quad \text{where} \quad g_i(\mathbf{x}) = \sigma_i(\mathbf{W}_i \mathbf{x} + \mathbf{c}_i)\,, \tag{3.5}$$

where C_g is taken to be the composition operator with g, each σ_i is a nonlinear activation function, and $\boldsymbol{\theta} = \{\mathbf{W}_i, \mathbf{c}_i\}_{i=1}^{l}$ denotes the neural network's parameterization. A large collection of activation functions exist in the literature, with the standard choice for deep networks being the rectified linear unit (ReLU) [54]. For interpolating the vector field, we

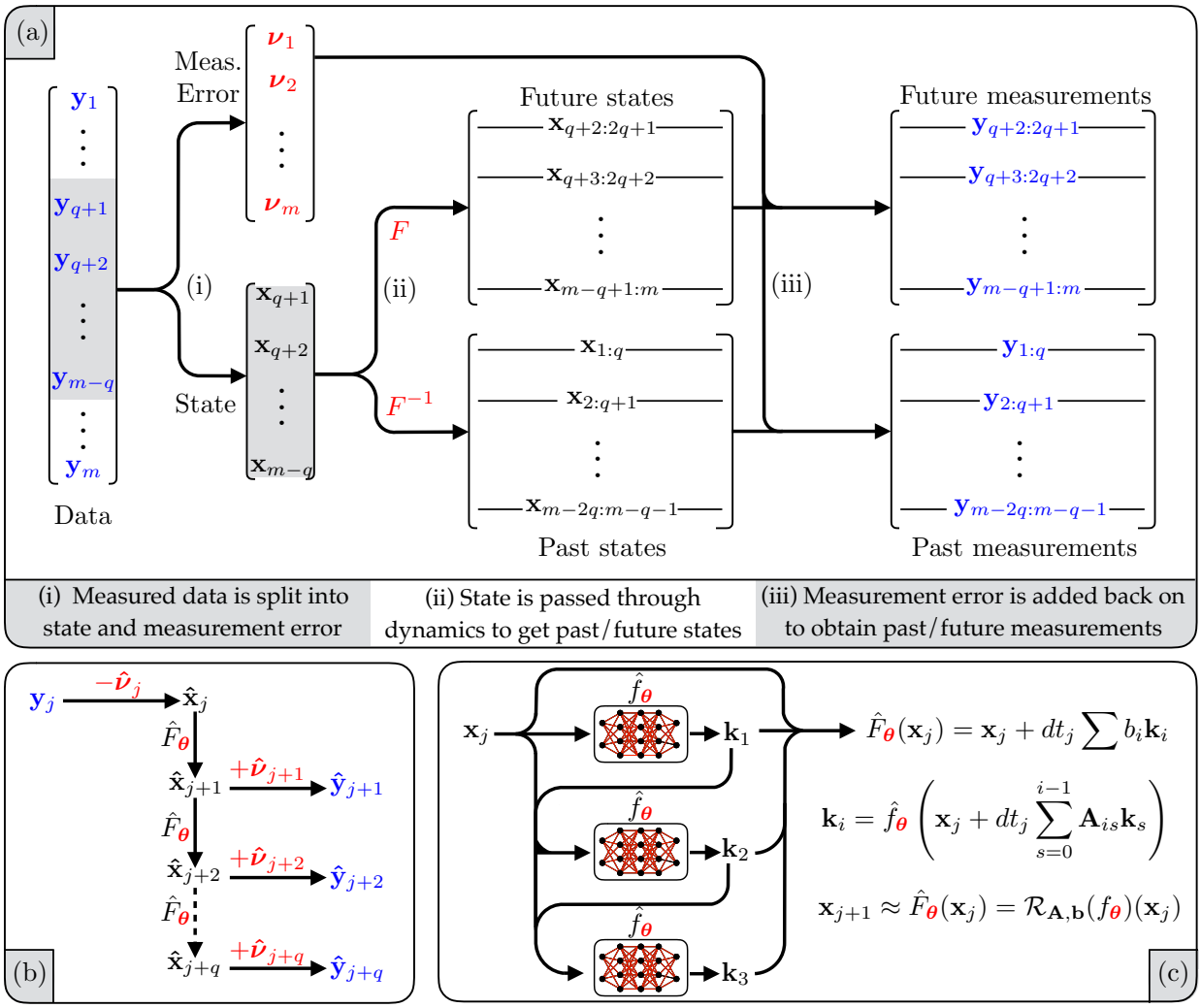


Figure 3.2: Schematic of de-noising framework for constrained learning of nonlinear system identification. **(a)** Flow of full dataset through (3.4) to find forward and backward measurements. Measurements are decomposed into learned measurement error and state, the state is propagated forward and backward using learned dynamics, and measurement error is added back to find forward and backward measurements and evaluate model accuracy. **(b)** Example of data-driven estimate of forward propagation of single observation from graph in panel a. **(c)** Example 3-step Runge-Kutta scheme to approximate the map F and isolate the vector field, which is modeled by a feed-forward neural network.

prefer a smooth activation function and instead use the exponential linear unit, given by

$$\sigma(\mathbf{x}) = \begin{cases} e^{\mathbf{x}} - 1, & \text{for } \mathbf{x} \leq 0 \\ \mathbf{x}, & \text{for } \mathbf{x} > 0 \end{cases} \tag{3.6}$$

evaluated component-wise for $\sigma_1, \ldots, \sigma_{l-1}$, and the identity for σ_l. Neural networks are typically trained by passing in known pairs of inputs and outputs, but doing so for $\hat{f}_{\boldsymbol{\theta}}$ would require robust approximations of the time derivative from noisy time series, which can be prohibitively difficult.

To avoid numerical differentiation of noisy data, we embed $\hat{f}_{\boldsymbol{\theta}}$ into a time-stepper to obtain a discrete map approximating the flow of the dynamical system, (3.3). We use explicit Runge-Kutta schemes because they allow us to make predictions from a single state measurement and are easily formed as an extension of the neural network for $\hat{f}_{\boldsymbol{\theta}}$, shown in Fig. 3.2 (c). Runge-Kutta schemes for autonomous systems are uniquely defined by weights $\mathbf{A} \in \mathbb{R}^{p \times p}, \mathbf{b} \in \mathbb{R}^p$, where p the the number of intermediate steps [88]. Given $\mathbf{A}, \mathbf{b}$, we let $\mathcal{R}_{\mathbf{A},\mathbf{b}}$ denote the operator induced by a time-stepper with parameters $\mathbf{A}, \mathbf{b}$ mapping a vector field to a discrete flow map. Applying $\mathcal{R}_{\mathbf{A},\mathbf{b}}$ to an approximation of the vector field $\hat{f}_{\boldsymbol{\theta}}$ gives an approximate flow map,

$$\hat{\mathbf{x}}_{j+1} = \hat{F}_{\boldsymbol{\theta}}(\mathbf{x}_j) = \mathcal{R}_{\mathbf{A},\mathbf{b}}(\hat{f}_{\boldsymbol{\theta}})(\mathbf{x}_j) \approx F(\mathbf{x}_j). \tag{3.7}$$

Incorporating estimates for the measurement error at each timestep extends (3.7) to a data-driven map from the observation $\mathbf{y}_j$ to $\mathbf{y}_{j+i}$, approximating the exact map in (3.4):

$$\hat{\mathbf{y}}_{j+i} = \hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+i}. \tag{3.8}$$

We have introduced model parameters $\boldsymbol{\theta}$ and $\hat{\mathbf{N}}$. It is common practice in machine learning to use the ℓ^2 error metric when training dynamics models [134, 117, 126], which may be interpreted as a maximum likelihood estimate given Gaussian errors. In our case this

is $\mathbf{y}_{j+1} - \hat{\mathbf{y}}_{j+1} = \boldsymbol{\xi}_j \sim \mathcal{N}(0, \sigma_{\xi^2}\mathbf{I})$ and maximum-likelihood estimation yields the following cost function,

$$\mathcal{L}_{MLE}(\boldsymbol{\theta}, \hat{\mathbf{N}}) = \sum_{j=1}^{m-1} \left\| \mathbf{y}_{j+1} - \left(\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+1} \right) \right\|_2^2. \tag{3.9}$$

The cost function in (3.9) has a global minimum with the trivial solution $\hat{f}_\theta = 0$ and estimated measurement error $\boldsymbol{\nu}_i = \mathbf{y}_i$ accounting for all observed dynamics. This minimum is not unique since many parameterizations $\boldsymbol{\theta}$ yield $\hat{f}_{\boldsymbol{\theta}} = 0$, and other trivial minima exist for constant $\hat{f}_{\boldsymbol{\theta}} = c$. In practice, locally minimizing solutions to (3.9) are nontrivial but also do not result in accurate models.

The inadequacy of (3.9) likely stems from the underdetermined nature of the problem and may be remedied by priors on the learned quantities that add regularization terms. A more robust cost function may be derived a a *maximum a posteriori* (MAP) estimate give a few assumptions on the timestepper error, measurement error, and neural network parameters. Most generally, these can be written as

$$\begin{aligned} \boldsymbol{\xi}_j &= \mathbf{x}_{j+1} - \hat{F}_{\boldsymbol{\theta}}(\mathbf{x}_j) \sim \pi_\xi \\ \boldsymbol{\nu}_j &= \mathbf{y}_j - \mathbf{x}_j \sim \pi_\nu \\ \mathbf{W}_{i,jk} &\sim \pi_w = \mathcal{N}(0, \sigma_w^2) \end{aligned} \tag{3.10}$$

where we have assumed absolutely continuous errors, so that π_ξ and π_ν are distribution functions and that all errors are independent. The timestepper error $\boldsymbol{\xi}$ may be interpreted as stochastic process noise or as error in the neural network interpolation of the dynamics. The former case would motivate also fitting an appropriate spatially dependent diffusion term and is not considered in this work since we are only concerned with deterministic dynamics. Let $\mathbf{X} = \mathbf{Y} - \mathbf{N}$ so that $\pi_{\mathbf{X}|\boldsymbol{\theta}}(\mathbf{Y} - \mathbf{N}|\boldsymbol{\theta}) = \pi_{\mathbf{Y}|\mathbf{N},\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{N}, \boldsymbol{\theta})$. The posterior

likelihood $\pi_{\mathbf{N},\boldsymbol{\theta}|\mathbf{Y}}(\mathbf{N},\boldsymbol{\theta}|\mathbf{Y})$ is,

$$\begin{aligned}\pi(\mathbf{N},\boldsymbol{\theta}|\mathbf{Y}) &= \frac{\pi(\mathbf{Y}|\mathbf{N},\boldsymbol{\theta})\pi(\mathbf{N},\boldsymbol{\theta})}{\pi(\mathbf{Y})} \\ &\propto \pi(\mathbf{Y}|\mathbf{N},\boldsymbol{\theta})\pi(\mathbf{N})\pi(\boldsymbol{\theta}) \\ &= \pi(\mathbf{X}|\boldsymbol{\theta})\pi(\mathbf{N})\pi(\boldsymbol{\theta}),\end{aligned} \tag{3.11}$$

where we have omitted subscripts for notational convenience. The evidence term $\pi(\mathbf{Y})$ is independent of $\mathbf{N}$ and $\boldsymbol{\theta}$ and it may be ignored in algorithms for estimating the latter two quantities. We examine terms in the numerator individually.

$$\begin{aligned}\pi(\mathbf{X}|\boldsymbol{\theta}) &= \pi(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m|\boldsymbol{\theta}) \\ &= \pi\left(\mathbf{x}_m|\boldsymbol{\theta},\{\mathbf{x}_i\}_{i=1}^{m-1}\right)\pi\left(\mathbf{x}_{m-1}|\boldsymbol{\theta},\{\mathbf{x}_i\}_{i=1}^{m-2}\right)\ldots\pi\left(\mathbf{x}_2|\boldsymbol{\theta},\mathbf{x}_1\right)\pi\left(\mathbf{x}_1|\boldsymbol{\theta}\right) \\ &= \pi(\mathbf{x}_1)\prod_{j=1}^{m-1}\pi_\xi\left(\mathbf{x}_{j+1}-\hat{F}_{\boldsymbol{\theta}}(\mathbf{x}_j)\right),\end{aligned} \tag{3.12}$$

where $\pi\left(\mathbf{x}_{j+1}|\boldsymbol{\theta},\{\mathbf{x}_i\}_{i=1}^{j}\right) = \pi\left(\mathbf{x}_{j+1}|\boldsymbol{\theta},\mathbf{x}_j\right)$ given the Markovian assumption in equation (3.10). The remaining terms are,

$$\begin{aligned}\pi(\mathbf{N}) &= \prod_{j=1}^{m}\pi_\nu(\boldsymbol{\nu}_j), \\ \pi(\boldsymbol{\theta}) &= \pi\left(\{\mathbf{W}_i,\mathbf{c}_i\}_{i=1}^{l}\right) \propto \prod_{i=1}^{l} e^{-\frac{1}{2}\sigma_w^{-2}\|\mathbf{W}_i\|_F^2}.\end{aligned} \tag{3.13}$$

Therefore,

$$\begin{aligned}\pi(\mathbf{N},\boldsymbol{\theta}|\mathbf{Y}) &\propto P(\mathbf{x}_1)\left(\prod_{j=1}^{m-1}\pi_\xi(\mathbf{x}_{j+1}-\hat{F}_{\boldsymbol{\theta}}(\mathbf{x}_j))\right)\left(\prod_{j=1}^{m}\pi_\nu(\boldsymbol{\nu}_j)\right)\left(\prod_{i=1}^{l}e^{-\frac{1}{2}\sigma_w^{-2}\|\mathbf{W}_i\|_F^2}\right) \\ &= P(\mathbf{x}_1)\left(\prod_{j=1}^{m-1}\pi_\xi(\mathbf{y}_{j+1}-\boldsymbol{\nu}_{j+1}-\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j-\boldsymbol{\nu}_j))\right)\left(\prod_{j=1}^{m}\pi_\nu(\boldsymbol{\nu}_j)\right)\left(\prod_{i=1}^{l}e^{-\frac{1}{2}\sigma_w^{-2}\|\mathbf{W}_i\|_F^2}\right).\end{aligned} \tag{3.14}$$

The term $\pi(\mathbf{x}_1)$ is relevant if there is a background density function for the initial condition. In this work we assume no prior knowledge of $\mathbf{x}_1$ and omit the term. Taking the negative log of the posterior likelihood we find,

$$\begin{aligned} L(\boldsymbol{\theta}, \mathbf{N}) &= -\ln(\pi(\mathbf{N}, \boldsymbol{\theta}|\mathbf{Y})) \\ &= -\sum_{j=1}^{m-1} \ln(\pi_\xi(\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j - \boldsymbol{\nu}_j) - \mathbf{y}_{j+1} + \boldsymbol{\nu}_{j+1})) - \sum_{j=1}^{m} \ln(\pi_\nu(\boldsymbol{\nu}_j)) + \sum_{i=1}^{l} \frac{1}{2\sigma^2} \|\mathbf{W}_i\|_F^2. \end{aligned} \tag{3.15}$$

If π_ξ and π_ν are taken to be Gaussian with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$, respectively, then the above cost function simplifies to,

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{N}) = \sum_{j=1}^{m-1} \frac{1}{2} \|\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j - \boldsymbol{\nu}_j) - \mathbf{y}_{j+1} + \boldsymbol{\nu}_{j+1}\|_{\mathbf{Q}^{-1}}^2 + \sum_{j=1}^{m} \frac{1}{2} \|\boldsymbol{\nu}_j\|_{\mathbf{R}^{-1}}^2 + \sum_{i=1}^{l} \frac{1}{2\sigma^2} \|\mathbf{W}_i\|_F^2. \tag{3.16}$$

If $\mathbf{Q}$ and $\mathbf{R}$ are unknown then they may be approximated using the ensemble Kalman filter after an initial estimate of the dynamics [10]; however we take a more naive approach and simply let them be multiples of the identity. Letting $\mathbf{Q} = \sigma_\xi^2 \mathbf{I}$ and $\mathbf{R} = \sigma_\nu^2 \mathbf{I}$ then,

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \mathbf{N}) &= \sum_{j=1}^{m-1} \frac{1}{2\sigma_\xi^2} \|\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j - \boldsymbol{\nu}_j) - \mathbf{y}_{j+1} + \boldsymbol{\nu}_{j+1}\|_2^2 + \sum_{j=1}^{m} \frac{1}{2\sigma_\nu^2} \|\boldsymbol{\nu}_j\|_2^2 + \sum_{i=1}^{l} \frac{1}{2\sigma^2} \|\mathbf{W}_i\|_F^2 \\ &\propto \sum_{j=1}^{m-1} \|\hat{F}_{\boldsymbol{\theta}}(\mathbf{y}_j - \boldsymbol{\nu}_j) - \mathbf{y}_{j+1} + \boldsymbol{\nu}_{j+1}\|_2^2 + \gamma \|\mathbf{N}\|_F^2 + \sum_{i=1}^{l} \beta \|\mathbf{W}_i\|_F^2 \end{aligned} \tag{3.17}$$

where $\gamma = \sigma_\xi^2 \sigma_\nu^{-2}$ and $\beta = \sigma_\xi^2 \sigma_w^{-2}$.

The derivation of (3.17) factors the likelihood of $\mathbf{X}$ into a likelihood over individual increments given by $\pi_\xi(\mathbf{x}_{j+1}|\mathbf{x}_j) = \pi_\xi(\boldsymbol{\xi}_j)$, removing any consideration of accuracy over multiple steps. The fact that $\boldsymbol{\xi}_j \neq 0$ even for deterministic systems (3.1), is a consequence of discretization error. However, many sequence models and data-assimilation methods use predictions over multiple timesteps. This is particularly notable in recurrent neural networks (RNN) [167, 118, 97, 121, 120, 126, 93] and variational data assimilation [34].

Blending the probabilistic framework, given by (3.17), with RNN based approaches to sequence models, we can generalize (3.17) to include predictions over multiple steps. The resulting cost function is given by,

$$\mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) = \sum_{j=q+1}^{m-q} \sum_{i=-q}^{q} \omega_i \left\| \hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+i} - \mathbf{y}_{j+i} \right\|_2^2 + \gamma \|\hat{\mathbf{N}}\|_F^2 + \beta \sum_{i=1}^{l} \|\mathbf{W}_i\|_F^2. \tag{3.18}$$

Note that (3.18) does not evaluate fidelity of prediction beyond a certain number of steps, given by the parameter q. Summing the error in (3.18) over all pairs of observations will result in a computationally stiff and intractable optimization problem. Indeed, figure 3.13 shows computational time to achieve a fixed threshold for relative change in 3.18 grows linearly with q. Moreover, chaotic dynamics will render observations statistically uncorrelated as they separate in time. We also weight each pair with exponentially decreasing importance in $|i|$, given by ω_i, according to the expected accumulation of error given inaccuracies in $\hat{\boldsymbol{\nu}}_j$.

In some cases, quantities may be measured in units that lead to disparate scales between variables. Normalization may be used to scale errors (for example by the precision matrix of the observations) so that terms in (3.18) do not disproportionately penalize certain directions. Doing so implicitly assumes that measurement noise is proportional to the variance of the observed data. If this is too strong of an assumption, then an algorithm that alternates between approximating noise covariance and the proposed method may be used, but this is beyond the scope of the current work.

Neural network parameters, as well as explicit estimates of the measurement error, are obtained by minimizing (3.18) using the quasi-Newton method L-BFGS-B [183] implemented in SciPy.

3.2.3 *Expected error and structure of loss function*

There is one last aspect of the computational framework we need to address. Equation (3.18) contains parameters ω_i weighting flow map error. The cost function evaluates the accuracy of a data-driven model against measured data using pairs of data separated by $i \in [1, q]$ timesteps. For larger i, we expect errors to accumulate, resulting in less accurate predictions. During the training procedure, error in long term predictions $|i| \gg 1$ may result in exploding gradients [119]. Therefore, we use an exponential weight ω_i to discount larger i. Here we estimate the error for an i timestep prediction and show that under a relaxed set of assumptions, an upper bound is exponential in i, justifying our choice of ω_i. Let $\boldsymbol{\eta}_{j,i}$ be the error in approximating $\mathbf{y}_{j+i}$ from $\mathbf{y}_j$. Then,

$$\boldsymbol{\eta}_{j,i} = \left(\hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+1}\right) - \mathbf{y}_{j+i}. \tag{3.19}$$

We are interesting in obtaining plausible estimates for the rate of growth of $\boldsymbol{\eta}_{j,i}$ as i grows from 1 to q. Let $\boldsymbol{\epsilon}_j$ and G denote the error in approximating the measurement noise and flow map respectively:

$$\begin{aligned} \hat{\boldsymbol{\nu}}_j - \boldsymbol{\nu}_j &= \boldsymbol{\epsilon}_j \\ \hat{F}_{\boldsymbol{\theta}} - F &= G. \end{aligned} \tag{3.20}$$

We restrict our attention to a domain $\mathcal{D} \subset \mathbb{R}^n$ and let $\alpha = \sup_{\mathbf{x}\in\mathcal{D}} \|\mathbf{D}F(\mathbf{x})\|$ where $\mathbf{D}F$ is the Jacobian, $\tilde{\alpha} = \sup_{\mathbf{x}\in\mathcal{D}} \|\mathbf{D}\hat{F}_{\boldsymbol{\theta}}(\mathbf{x})\|$, and $\zeta = \sup_{\mathbf{x}\in\mathcal{D}} \|G(\mathbf{x})\|$. We assume that the data-driven model is sufficiently accurate that $\zeta \ll 1$ and $\|\boldsymbol{\epsilon}_j\| < \mu \ll 1$. The error in predicting $\mathbf{y}_{j+i}$ from $\mathbf{y}_j$ is,

$$\begin{aligned} \boldsymbol{\eta}_{j,i} &= (\hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+1}) - \mathbf{y}_{j+1} \\ &= (\hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \boldsymbol{\nu}_j - \boldsymbol{\epsilon}_j) + \boldsymbol{\nu}_{j+i} + \boldsymbol{\epsilon}_{j+i}) - \mathbf{y}_{j+i} \\ &= \hat{F}^i_{\boldsymbol{\theta}}(\mathbf{x}_j - \boldsymbol{\epsilon}_j) - \mathbf{x}_{j+i} + \boldsymbol{\epsilon}_{j+1} \\ &= \hat{F}^i_{\boldsymbol{\theta}}(\mathbf{x}_j) - \mathbf{x}_{j+i} + \boldsymbol{\epsilon}_{j+1} + \mathcal{O}\left(\tilde{\alpha}^i \|\boldsymbol{\epsilon}_j\|\right) \\ &= (F+G)^i(\mathbf{x}_j) - F^i(\mathbf{x}_j) + \mathcal{O}\left((\tilde{\alpha}^i + 1)\mu\right). \end{aligned} \tag{3.21}$$

Focusing on the term $(F+G)^i(\mathbf{x}_j)$, let $\boldsymbol{\delta}_k = G((F+G)^{k-1}(\mathbf{x}_i))$ be the error in the data-driven flow map evaluated at $\hat{F}_{\boldsymbol{\theta}}^{k-1}(\mathbf{x}_j)$. Then $\|\boldsymbol{\delta}_k\| \leq \zeta$ and,

$$\begin{aligned}
(F+G)(\mathbf{x}) &= F(\mathbf{x}) + \boldsymbol{\delta}_1 \\
(F+G)^2(\mathbf{x}) &= (F+G)(F(\mathbf{x}) + \boldsymbol{\delta}_1) \\
&\approx F^2(\mathbf{x}) + \mathbf{D}F(F(\mathbf{x}))\boldsymbol{\delta}_1 + \boldsymbol{\delta}_2 + \mathcal{O}(\zeta^2) \\
(F+G)^3(\mathbf{x}) &\approx (F+G)(F^2(\mathbf{x}) + \mathbf{D}F(F(\mathbf{x}))\boldsymbol{\delta}_1 + \boldsymbol{\delta}_2) \\
&= F^3(\mathbf{x}) + \mathbf{D}F(F^2(\mathbf{x}))\mathbf{D}F(F(\mathbf{x}))\boldsymbol{\delta}_1 + \mathbf{D}F(F^2(\mathbf{x}))\boldsymbol{\delta}_2 + \boldsymbol{\delta}_3 + \mathcal{O}(\zeta^2).
\end{aligned} \tag{3.22}$$

Continuing to higher powers of $(F+G)$ we find,

$$\begin{aligned}
(F+G)^i(\mathbf{x}) - F(\mathbf{x}) &= \sum_{k=1}^{i} \left(\prod_{l=1}^{i-k} \mathbf{D}F(F^l(\mathbf{x})) \right) \boldsymbol{\delta}_k \\
\|(F+G)^i(\mathbf{x}) - F(\mathbf{x})\| &\leq \sum_{k=1}^{i} \left\| \left(\prod_{l=1}^{i-k} \mathbf{D}F(F^l(\mathbf{x})) \right) \boldsymbol{\delta}_k \right\| \\
&\leq \sum_{k=0}^{i-1} \zeta \alpha^k \\
&= \frac{\zeta(\alpha^i - 1)}{\alpha - 1},
\end{aligned} \tag{3.23}$$

for $\alpha \neq 1$, and $(F+G)^i(\mathbf{x}) - F(\mathbf{x}) \leq i\zeta$ otherwise. Having ignored all quadratic terms in μ and ζ, we find that an upper bound $\boldsymbol{\eta}_{j,i}$ is given by,

$$\|\boldsymbol{\eta}_{j,i}\| \leq \frac{\zeta(\alpha^i - 1)}{\alpha - 1} + \left(\tilde{\alpha}^i + 1\right)\mu, \tag{3.24}$$

where the norm used is the same as in the definitions of α, $\tilde{\alpha}$, ζ, and μ. This expression is exponential in i. A similar expression may be derived for $i < 0$ by following the same steps using the inverse flow maps. Therefore we expect error to grow exponentially in the number of forward and backward timesteps. In practice it would be very difficult to precisely estimate the quantity given in (3.24), so we assume $\boldsymbol{\eta}_i = \boldsymbol{\eta}_0 \rho^i$ for some $\rho > 1$.

In particular, for chaotic systems and longer timesteps, one would use larger values of ρ, resulting in a more aggressive exponential discount in ω_i.

The accumulation of error through multiple iterations of the flow map informs how we weight our loss function. In many machine learning tasks the canonical choice for loss function is mean square error, which is derived from the maximum likelihood estimate given an assumption of Gaussian distributed errors with identity covariance matrix. For this work, we use the ℓ^2 metric for our loss function and weight errors for i-step predictions with exponentially decreasing magnitude $\omega_i = \omega_0 \rho^{-i}$ and $\rho^{-1} = 0.9$. For non-chaotic problems, $\rho = 1$ may be appropriate, as is supported by the analysis shown in figure 3.16. This assumes Gaussian distributed error for all predictions, which is naive. However, any standard metric would fall short of capturing the true error of our computational framework for any reasonably complex dynamical system. Future work is required to investigate this more carefully.

3.2.4 Local Minima

An important consideration in any non-convex optimization problem is the existence of local minima and saddle points. Since (3.18) is non-convex, we cannot guarantee that any optimization algorithm will arrive at the global minimum. However, there is reason to believe that this is not a serious flaw in the method. Reference [29] shows that the likelihood of arriving at a high cost local minimum decreases rapidly with network size. Hence, with deeper architectures, any local minimum is more likely to be "good," in the sense of being low cost. Saddle points are discussed in [42] and are shown to be problematic for quasi-Newton type optimizers in high dimensions. The authors of [42] suggest a modified Newton type algorithm that affectively eliminates the saddle point problem, although implementing this method is beyond the scope of the current work.

A second consideration is non-convexity with respect to the estimation of measurement noise (equivalently the state). Consider, with fixed dynamics, the function $\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{N}) =$

$\mathcal{L}(\boldsymbol{\theta}, \mathbf{N})$. Since (3.18) passes estimates of measurement noise through the neural network, $\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{N})$ is non-convex and it may be possible, even with fixed dynamics, for the optimizer to settle into a local minimum. Careful analysis of the landscape of (3.18) and $\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{N})$ is beyond the scope of this work but regularization of $\hat{\mathbf{N}}$ in (3.18) does provide some small guarantee that any local minima cannot be a shifted state. More specifically, any critical point $(\boldsymbol{\theta}, \hat{\mathbf{N}})$ of (3.18) has sample mean $\bar{\boldsymbol{\nu}} = \frac{1}{m}\sum \hat{\boldsymbol{\nu}}_j = 0 \in \mathbb{R}^n$.

It suffices to show that any $(\boldsymbol{\theta}, \hat{\mathbf{N}})$ with $\bar{\boldsymbol{\nu}} \neq 0$ results in a nonzero gradient of (3.18). Consider the direction vector $\mathbf{u} = (\mathbf{u}_{\boldsymbol{\theta}}, \mathbf{u}_{\mathbf{N}})$ given by,

$$\begin{aligned} \mathbf{u}_{\boldsymbol{\theta}} &= \{(\mathbf{u}_{\mathbf{W}_1}, \mathbf{u}_{\mathbf{c}_1}), (\mathbf{u}_{\mathbf{W}_2}, \mathbf{u}_{\mathbf{c}_2}), \ldots, (\mathbf{u}_{\mathbf{W}_l}, \mathbf{u}_{\mathbf{c}_l})\} \\ &= \left\{(0, \frac{-\bar{\boldsymbol{\nu}}}{\sqrt{m+1}\|\bar{\boldsymbol{\nu}}\|_2}), (0,0), \ldots, (0,0)\right\} \\ \mathbf{u}_{\mathbf{N}} &= \left[\frac{-\bar{\boldsymbol{\nu}}}{\sqrt{m+1}\|\bar{\boldsymbol{\nu}}\|_2}, \ldots, \frac{-\bar{\boldsymbol{\nu}}}{\sqrt{m+1}\|\bar{\boldsymbol{\nu}}\|_2}\right]. \end{aligned} \tag{3.25}$$

Consider an ϵ perturbation of $(\boldsymbol{\theta}, \hat{\mathbf{N}})$ by $\mathbf{u}$ and let $\epsilon' = \epsilon(m+1)^{-1/2}\|\bar{\boldsymbol{\nu}}\|_2^{-1}$ so that,

$$\begin{aligned} \boldsymbol{\theta}' &= \boldsymbol{\theta} + \epsilon \mathbf{u}_{\boldsymbol{\theta}} = \{(\mathbf{W}_1, \mathbf{c}_1 - \epsilon'\bar{\boldsymbol{\nu}}), (\mathbf{W}_2, \mathbf{c}_2), \ldots, (\mathbf{W}_l, \mathbf{c}_l)\} \\ \hat{\mathbf{N}}' &= \hat{\mathbf{N}} + \epsilon \mathbf{u}_{\mathbf{N}} = [\hat{\boldsymbol{\nu}}_1 - \epsilon'\bar{\boldsymbol{\nu}}, \ldots, \hat{\boldsymbol{\nu}}_m - \epsilon'\bar{\boldsymbol{\nu}}]. \end{aligned} \tag{3.26}$$

From (3.5) we know $\hat{f}_{\boldsymbol{\theta}'}(\mathbf{x}) = \hat{f}_{\boldsymbol{\theta}}(\mathbf{x} - \epsilon'\bar{\boldsymbol{\nu}})$ and therefore $\hat{F}_{\boldsymbol{\theta}'}(\mathbf{x}) = \hat{F}_{\boldsymbol{\theta}}(\mathbf{x} - \epsilon'\bar{\boldsymbol{\nu}}) + \epsilon'\bar{\boldsymbol{\nu}}$. Evalu-

ating (3.18) at $(\boldsymbol{\theta}', \hat{\mathbf{N}}')$ gives,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}', \hat{\mathbf{N}}') &= \sum_{j=q+1}^{m-q} \sum_{i=-q}^{q} \omega_i \left\| \hat{F}^i_{\boldsymbol{\theta}'}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}'_j) + \hat{\boldsymbol{\nu}}'_{j+i} - \mathbf{y}_{j+i} \right\|_2^2 + \gamma \|\hat{\mathbf{N}}'\|_F^2 + \beta \sum_{i=1}^{l} \|\mathbf{W}_i\|_F^2 \\
&= \sum_{j=q+1}^{m-q} \sum_{i=-q}^{q} \omega_i \left\| \hat{F}^i_{\boldsymbol{\theta}'}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j + \epsilon' \bar{\boldsymbol{\nu}}) + \hat{\boldsymbol{\nu}}_{j+i} - \epsilon' \bar{\boldsymbol{\nu}} - \mathbf{y}_{j+i} \right\|_2^2 \\
&\quad + \gamma \sum_{j=1}^{m} \|\hat{\boldsymbol{\nu}}_j - \epsilon' \bar{\boldsymbol{\nu}}\|_2^2 + \beta \sum_{i=1}^{l} \|\mathbf{W}_i\|_F^2 \\
&= \sum_{j=q+1}^{m-q} \sum_{i=-q}^{q} \omega_i \left\| \hat{F}^i_{\boldsymbol{\theta}}(\mathbf{y}_j - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\nu}}_{j+i} - \mathbf{y}_{j+i} \right\|_2^2 + \gamma \sum_{j=1}^{m} \|\hat{\boldsymbol{\nu}}_j - \epsilon' \bar{\boldsymbol{\nu}}\|_2^2 + \beta \sum_{i=1}^{l} \|\mathbf{W}_i\|_F^2 \\
&= \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) + \gamma \sum_{j=1}^{m} \left(\epsilon'^2 \|\bar{\boldsymbol{\nu}}\|_2^2 - 2\epsilon' \langle \hat{\boldsymbol{\nu}}_j, \bar{\boldsymbol{\nu}} \rangle \right) \\
&= \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) + m\gamma(\epsilon'^2 - 2\epsilon') \|\bar{\boldsymbol{\nu}}\|_2^2 \\
&= \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) - \frac{2m\gamma \|\bar{\boldsymbol{\nu}}\|_2 \epsilon}{\sqrt{m+1}} + \frac{2m\gamma\epsilon^2}{m+1}
\end{aligned} \tag{3.27}$$

The directional derivative $\nabla \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) \cdot \mathbf{u}$ is given by,

$$\begin{aligned}
\nabla \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}) \cdot \mathbf{u} &= \lim_{\epsilon \to 0} \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \mathbf{u}_{\boldsymbol{\theta}}, \hat{\mathbf{N}} + \epsilon \mathbf{u}_{\hat{\mathbf{N}}}) - \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}})}{\epsilon} \\
&= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left(-\frac{2m\gamma \|\bar{\boldsymbol{\nu}}\|_2 \epsilon}{\sqrt{m+1}} + \frac{2m\gamma\epsilon^2}{m+1} \right) \\
&= -\frac{2m\gamma \|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{m+1}}.
\end{aligned} \tag{3.28}$$

This suffices to show that any critical point of (3.18) must satisfy $\bar{\boldsymbol{\nu}} = 0$.

3.2.5 Measuring error

In addition to the loss function derived in Sec. 3.2.2, we use several other metrics to evaluate the accuracy of models produced by minimizing (3.18). It is possible to evaluate these metrics for the problems considered in this work because the equations and measurement

noise are both known. Although these are not generally available for a new dataset, they provide a quantitative basis for comparing performance with different quantities of noise, volumes of data, and timestepping schemes. To evaluate the accuracy of $\hat{f}_{\boldsymbol{\theta}}$, we use the relative squared ℓ^2 error between the true and data-driven vector fields,

$$E_f(\hat{f}_{\boldsymbol{\theta}}) = \frac{\sum_{j=1}^{m} \|f(\mathbf{x}_j) - \hat{f}_{\boldsymbol{\theta}}(\mathbf{x}_j)\|_2^2}{\sum_{j=1}^{m} \|f(\mathbf{x}_j)\|_2^2}. \tag{3.29}$$

Here the error is only evaluated along the noiseless training data. This results in substantially lower error than if the two vector fields were compared on a larger set away from the training data, as will be discussed in Sec. 3.4.

The other learned quantity is an estimate for the measurement noise $\boldsymbol{\nu}_j$, or equivalently the de-noised state $\mathbf{x}_j$, at each timestep. The mean ℓ^2 difference between the true and learned measurement error is,

$$E_N(\hat{\mathbf{N}}) = \frac{1}{m}\sum_{j=1}^{m} \|\boldsymbol{\nu}_j - \hat{\boldsymbol{\nu}}_j\|_2^2 = \frac{1}{m}\sum_{j=1}^{m} \|\mathbf{x}_j - \hat{\mathbf{x}}_j\|_2^2. \tag{3.30}$$

The ℓ^2 distance between the true state of the training data and the forward orbit of $\mathbf{x}_1$ as predicted by $\hat{F}_{\boldsymbol{\theta}}$ is computed as,

$$E_F(\hat{F}_{\boldsymbol{\theta}}, \mathbf{x}_1) = \frac{1}{\|\mathbf{X}\|_F^2}\sum_{j=1}^{m-1} \left\|\mathbf{x}_j - \hat{F}_{\boldsymbol{\theta}}^j(\mathbf{x}_1)\right\|_2^2. \tag{3.31}$$

The last error (3.31) is the most stringent and may be highly sensitive to small changes in the dynamics that reflect numerical error more than inaccuracies. E_F will not yield informative results for dynamics evolving on a chaotic attractor or slowly diverging from an unstable manifold. We therefore only consider it on the first example of a damped oscillator.

3.3 Results

In this section we test the performance of the methods discussed in Sec. 3.2 on a range of canonical problems of increasing complexity. To demonstrate robustness, we consider each problem with varying levels of corruption, meant to replicate the effects of measurement noise. In most cases, independent and identically distributed Gaussian noise is added to each component of the dataset with zero mean and amplitude equal to a given percent of the standard deviation of the data. That is,

$$\boldsymbol{\nu} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^2_{\mathbf{N}}), \qquad \boldsymbol{\Sigma}^2_{\mathbf{N}} = \left(\frac{\text{noise }\%}{100}\right)^2 \text{diag}(\boldsymbol{\Sigma}^2_{\mathbf{X}}), \tag{3.32}$$

where $\boldsymbol{\Sigma}^2_{\mathbf{X}}$ is the covariance of the true trajectory. However, even though we have used Gaussian error assumptions to motivate (3.18), the method is shown to be accurate on non-Gaussian noise distributions. For the Lorenz equation, the measurement noise is drawn from a Student's T distribution. In each case, an initial estimate for noise is obtained by a simple smoothing operation performed on the raw data. While not necessary, this was found to speed up the optimization routine.

The parameters $q = 10$, $\omega_i = 0.9^{|i|-1}$, $\gamma = 10^{-5}$, and $\beta = 10^{-8}$ were used on all but two of the examples shown and may be taken as a canonical choice. Parameters may be adjusted when the learned model diverges on training dataset or when $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$ is not sufficiently smooth. The double pendulum example in figure 3.12 for 5% and 10% noise use $\gamma = 10^{-6}$ and $\gamma = 10^{-7}$, respectively. Weights for the neural networks are initialized using the Xavier initialization [52] native to TensorFlow.

3.3.1 *Linear Example*

We first demonstrate the proposed method on a simple linear example with parameters $\epsilon = 0.05$ and $(b_1, b_2) = (1, -3)$, indicating a stable spiral:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\epsilon & -1 \\ 1 & -\epsilon \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{3.33}$$

A training dataset is constructed from a single trajectory using $m = 500$ timesteps and $dt = 0.02$. Figure 3.3 shows the results of applying the proposed method to the dataset using both 5% and 10% measurement noise, and with both a linear approximation of the dynamics and a 3 layer neural network. We show the learned vector field, error of the learned vector field, indicating regions of phase space where training data is available, and the learned measurement noise $\hat{\boldsymbol{\nu}}$ in the x-coordinate for a subset of the time series.

Figure 3.3 is particularly notable since the cases where a linear ansatz is used for $\hat{f}_{\boldsymbol{\theta}}$ both have substantially less error than the neural network. This is not surprising since the true dynamics may be exactly represented. Learned dynamics with the linear ansatz and 5% or 10% noise are given by,

$$\begin{aligned} \text{5 \% noise}: \quad \hat{f}_{\boldsymbol{\theta}} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} -0.050 & -1.000 \\ 0.999 & -0.050 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.992 \\ -2.995 \end{pmatrix} \\ \text{10 \% noise}: \quad \hat{f}_{\boldsymbol{\theta}} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} -0.058 & -1.001 \\ 0.998 & -0.043 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1.021 \\ -3.008 \end{pmatrix}. \end{aligned} \tag{3.34}$$

Error metrics for each setup used for the linear test problem are shown in table 3.1. We highlight that due to the nonconvexity of (3.18) these results may differ given different instantiations of the neural network and also depend on particular instantiations of measurement noise. However, the variance on error metrics are low for this example so values in table 3.1 may be understood as a good indicator of performance.

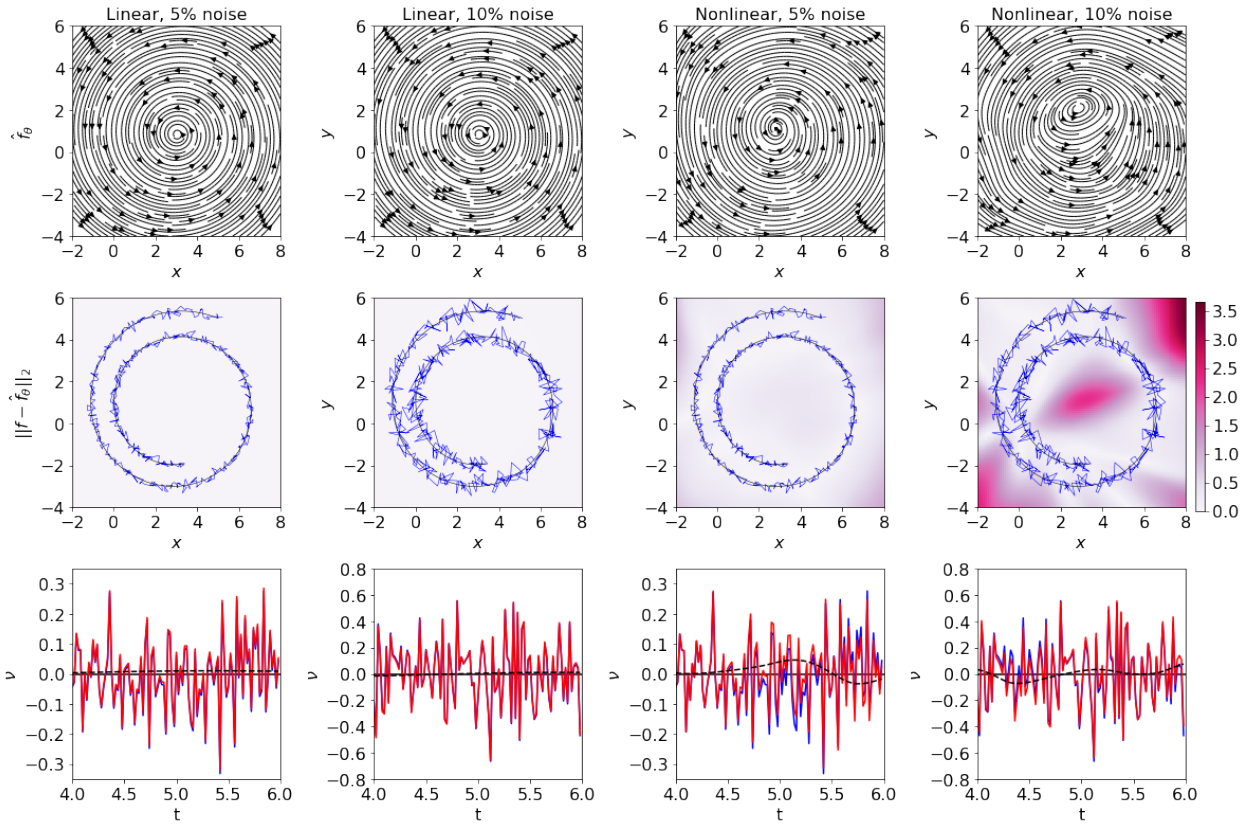


Figure 3.3: Results for stable spiral with several network architectures and noise magnitudes. Top row: Learned vector field $\hat{f}_{\boldsymbol{\theta}}$ for each configuration. Middle row: ℓ^2 error of learned vector field plotted with training data $\mathbf{Y}$ in blue and true state $\mathbf{X}$ in black. Bottom row: True x-component of measurement error $\boldsymbol{\nu}$ in blue and learned $\hat{\boldsymbol{\nu}}$ in red, with difference as dashed black line.

Table 3.1: Error for linear test case with varying levels of noise and neural network structures.

Setup	Linear, 5% noise	Linear, 10% noise	Nonlinear, 5% noise	Nonlinear, 10% noise
E_N	$2.846e-4$	$5.448e-4$	$1.440e-3$	$4.981e-3$
E_f	$5.702e-4$	$3.719e-4$	$2.736e-2$	$8.971e-2$
E_F	$3.443e-3$	$1.490e-3$	$1.379e-2$	$8.836e-2$

Table 3.2: Error for cubic oscillator model with varying noise.

% Noise	0	1	5	10	10, $dt \sim exp$	25
E_N	$3.077e-7$	$2.570e-6$	$3.561e-5$	$1.292e-4$	$1.297e-4$	$7.847e-4$
E_f	$5.187e-5$	$1.516e-4$	$7.692e-4$	$2.436e-3$	$2.724e-3$	$1.319e-2$
E_F	$1.150e-4$	$1.955e-3$	$3.735e-2$	0.2170	$5.124e-3$	0.6241

3.3.2 Cubic oscillator

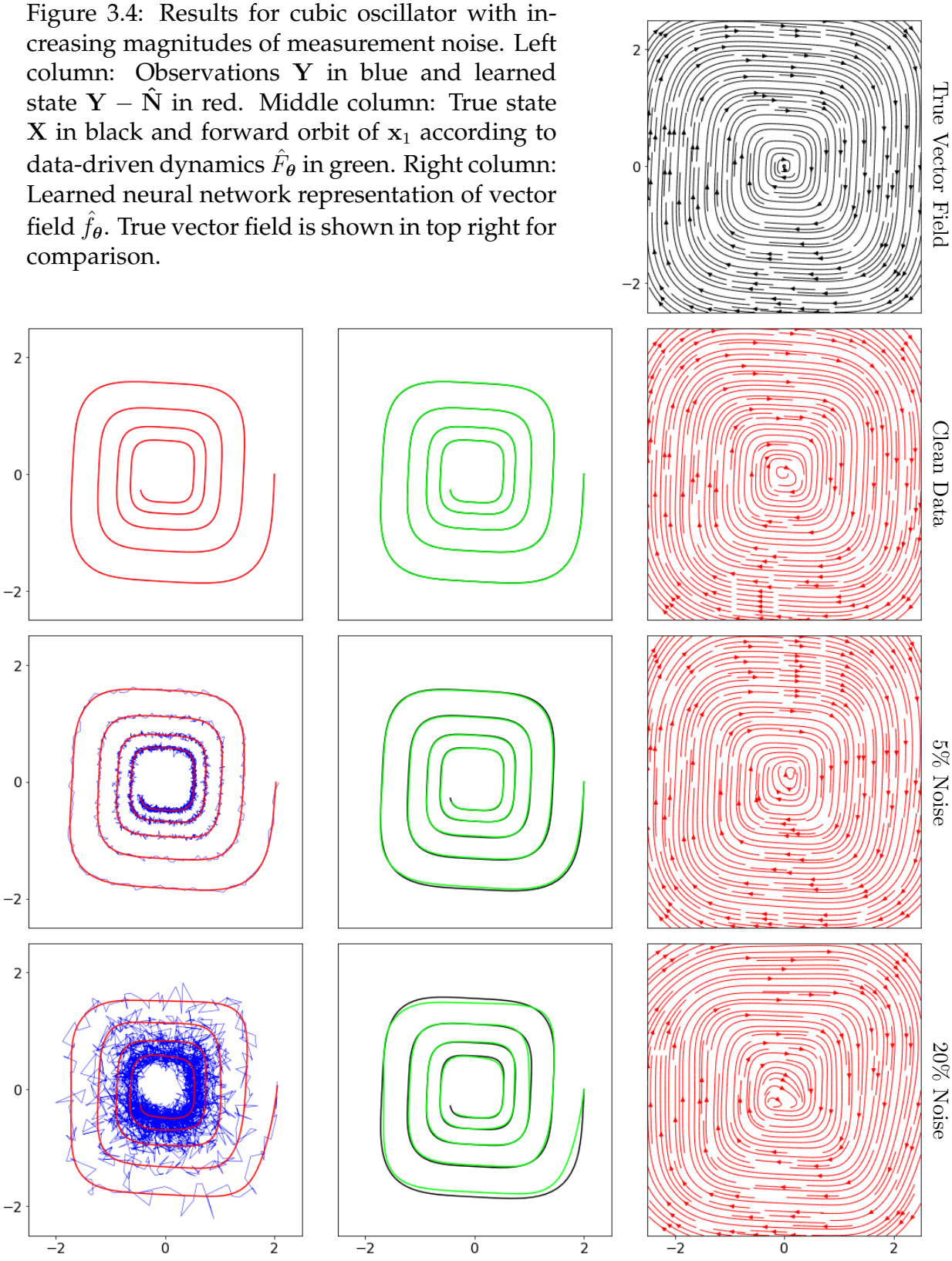
We consider the damped cubic oscillator, which has been used as a test problem for several data-driven methods to learn dynamical systems [18, 134]:

$$\begin{aligned} \dot{x} &= -0.1x^3 + 2y^3 \\ \dot{y} &= -2x^3 - 0.1y^3. \end{aligned} \tag{3.35}$$

We generate $2,500$ snapshots from $t = 0$ to $t = 25$ via high-fidelity simulation and then corrupt this data with varying levels of artificial noise, ranging from 0 to 20 percent. Models are trained by embedding a neural network with three hidden layers, containing 32 nodes each, in a four-step Runge-Kutta scheme. For each dataset, we obtain explicit estimates of the measurement noise and a neural network approximation of the vector field in (3.35), which is used to integrate a trajectory from the same initial condition to compare the relative error. Table 3.2 provides a summary of the error metrics from Sec. 3.2.5 evaluated across various noise levels. At higher noise levels, there is a substantial increase in E_F due to a phase shift in the reconstructed solution. We also tested the method on a dataset with random timesteps, drawn from an exponential distribution, $t_{j+1} - t_j \sim \exp(0.01)$. Error in approximating the vector field and noise were similar to the case with a constant timestep, while E_F was significantly lower. This suggests future work to perform a careful comparison between the cases of constant and variable timsteps.

Figure 3.4 shows the model predictions and vector fields for increasing amounts of measurement noise. In the left column, observations $\mathbf{Y}$ are plotted against the inferred

Figure 3.4: Results for cubic oscillator with increasing magnitudes of measurement noise. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ according to data-driven dynamics $\hat{F}_{\boldsymbol{\theta}}$ in green. Right column: Learned neural network representation of vector field $\hat{f}_{\boldsymbol{\theta}}$. True vector field is shown in top right for comparison.



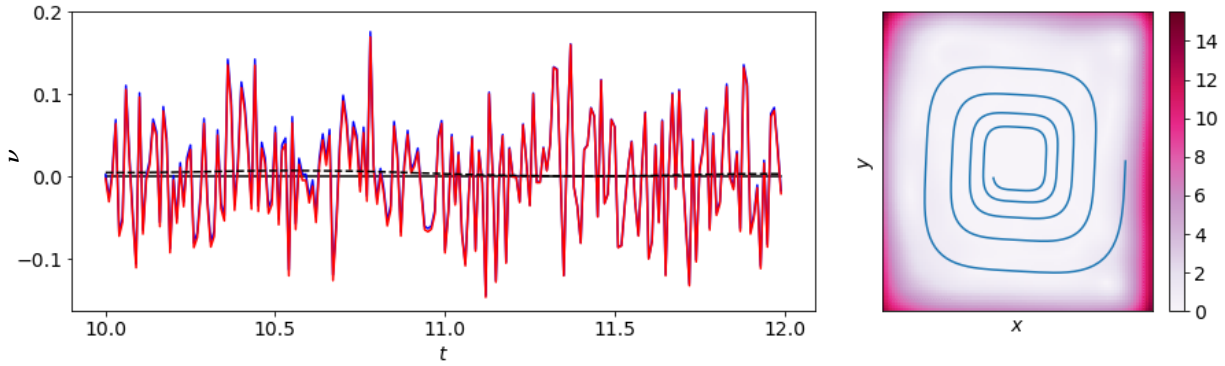


Figure 3.5: Results from a single trial of the cubic oscillator with 10% noise. Left: True measurement noise (blue), learned measurement noise (red), and error (dotted black line). Right: Heat map of vector field approximation error with the noiseless training trajectory plotted for reference. Note the significant increase in error magnitude in regions far from the training data.

state, $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$. The middle column shows the noiseless trajectory $\mathbf{X}$ alongside the forward orbit of $\mathbf{x}_1$ according to the learned timestepper $\hat{F}_{\boldsymbol{\theta}}$. The learned vector field $\hat{f}_{\boldsymbol{\theta}}$ for each noise level is plotted in the right column with the true vector field for reference. Results show that the proposed method is highly robust to significant measurement noise.

Figure 3.5 shows the error in the approximation of the measurement noise and the vector field for a single time series corrupted by 10% Gaussian measurement noise. The exact measurement noise in the x-coordinate is shown alongside the learned measurement noise for 200 timesteps. The ℓ^2 error in the approximation of the vector field is also shown with the uncorrupted training trajectory for a spatial reference. The vector field error is generally small near the training data and grows considerably near the edges of the domain.

Exact measures of error will vary depending on the particular instantiation of measurement noise and initialization of model parameters. Figure 3.6 shows the average error across fifty trials for each of the metrics discussed in Sec. 3.2.5. We compute averages for E_F using median rather than mean, since a single trajectory out of the fifty trials with 18% noise was divergent, resulting in non-numerical values.

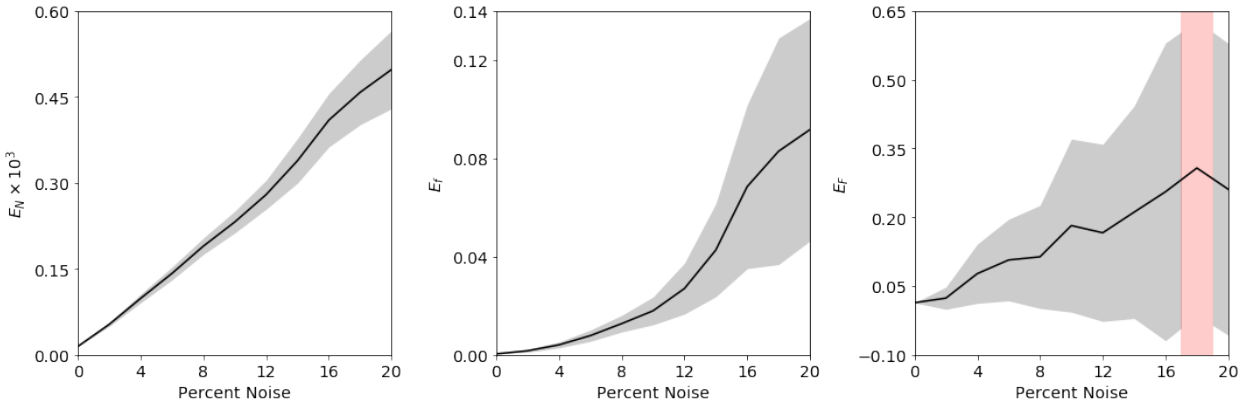


Figure 3.6: Average error over 50 trials and standard deviations for the cubic oscillator at varying levels of measurement noise. Left: Mean E_N and shaded region indicating one standard deviation. Center: Mean E_f and standard error. Right: Median normalized ℓ^2 difference between true trajectory and forward orbit under $\hat{f}_{\boldsymbol{\theta}}$, given by E_F. The distribution for 18% noise is omitted due to a single unstable trajectory resulting in non-numeric data; however, the median is reported.

3.3.3 Lorenz system

The next example is the Lorenz system, which originated as a simple model of atmospheric convection and became the canonical example for demonstrating chaotic behavior. We consider the Lorenz system with the standard parameter set $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z. \end{aligned} \tag{3.36}$$

The training dataset consists of a single trajectory with $2,500$ timesteps from $t = 0$ to $t = 25$ with initial condition $(x_0, y_0, z_0) = (5, 5, 25)$ starting near the attractor. The vector field f in (3.36) is modeled by a neural network with three hidden layers containing 64 nodes each, embedded in a four-step Runge Kutta scheme to approximate F. Results for several levels of measurement corruption, including noise drawn from a Student's T distribution, are shown in Fig. 3.7. Approximation errors for the Lorenz system at varying

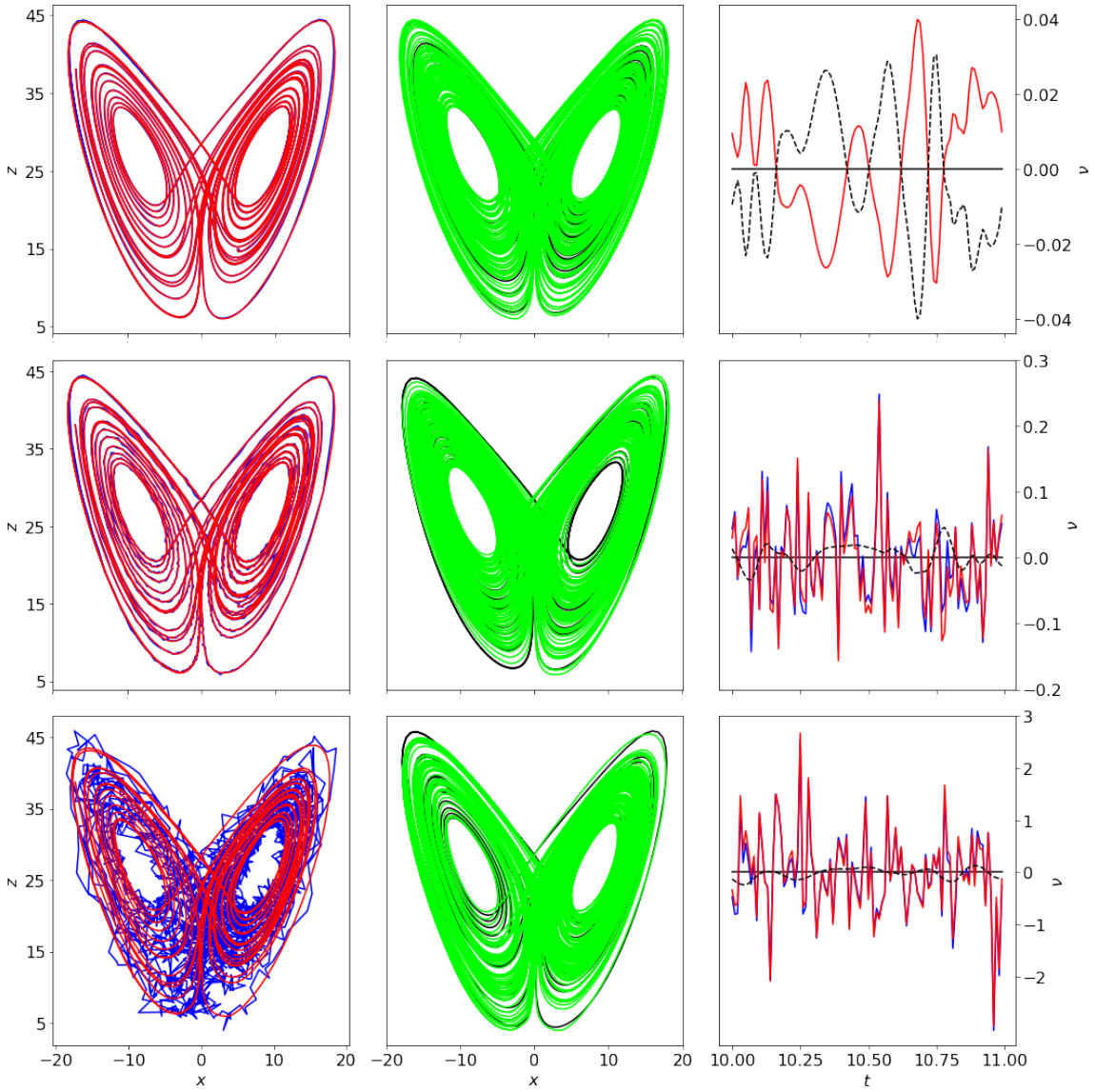


Figure 3.7: Results for Lorenz system with increasing magnitudes of measurement noise. Top row: Clean data. Second row: 1% Gaussian noise. Bottom row: 10% noise from Student's T distribution with 10 degrees of freedom. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. The prediction is extended to T_{max} five times that of training data. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in noise estimate in dashed black.

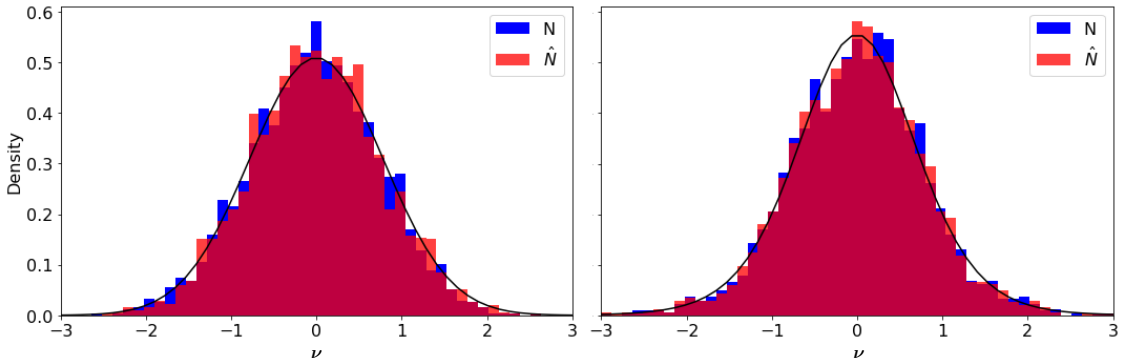


Figure 3.8: Histograms showing true and learned sample distribution of measurement noise with distribution of measurement noise plotted in black. Left: Learned noise from Lorenz system with 10% Gaussian distributed noise. Right: Learned noise from Lorenz system with 10% noise from Student's T distribution with 10 degrees of freedom.

levels of Gaussian distributed noise are summarized in table 3.3.

In many cases, it may be important to estimate the distribution of the measurement noise, in addition to the point-wise estimates. Figure 3.8 shows the true empirical measurement error distribution for the training data along with the distribution of the learned measurement error for the Lorenz system corrupted with either 10% Gaussian noise or 10% noise from a Student's T distribution with 10 degrees of freedom; the analytic distribution is also shown for reference. In both cases, the approximated error distribution faithfully captures the true underlying distribution of the measurement error. Mean, variance, skew, and excess kurtosis of the analytic, empirical, and learned distribution of measurement noise for the x-coordinate are shown in table 3.4.

Long term statistics for the Lorenz system model learned on a dataset of length $m =$

Table 3.3: Error for Lorenz system with varying noise.

% Noise	0	1	5	10	15
E_N	$4.722e-3$	$4.783e-3$	$2.670e-2$	$7.356e-2$	0.2248
E_f	$9.892e-4$	$9.361e-4$	$1.988e-3$	$2.299e-3$	$6.340e-3$

Table 3.4: Moments of analytic, empirical, and learned measurement noise in x-coordinate.

Gaussian Meas. Error					Student's T Meas. Error			
μ	σ^2	γ_1	κ		μ	σ^2	γ_1	κ
0	1.111	0	0		0	0.6143	0	1
-0.0145	0.5852	0.0171	-0.0920		-0.0192	0.6143	0.0150	0.6242
-0.0006	0.5794	0.0093	-0.0754		-0.0003	0.6055	-0.0633	0.7718

10000 with 5% noise are shown in Fig. 3.9. Trajectories shorter than $m = 5000$ often resulted in long term trajectories eventually diverging to a fixed point, rendering long term statistics highly inaccurate.

3.3.4 Low Reynolds number fluid flow past a cylinder

As a more complex example, we consider the high-dimensional data generated from a simulation of fluid flow past a circular cylinder at a Reynolds number of 100 based on cylinder diameter. Flow around a cylinder has been a canonical problem in fluid dynamics for decades. One particularly interesting feature of the flow is the presence of a Hopf bifurcation occurring at $Re = 47$, where the flow transitions from a steady configuration to laminar vortex shedding. The low-order modeling of this flow has a rich history, culminating in the celebrated mean-field model of Noack *et al.* [115], which used Galerkin projection and a separation of timescales to approximate the cubic Hopf nonlinearity with quadratic nonlinearities arising in the Navier-Stokes equations. This flow configuration has since been used to test nonlinear system identification [18, 90], and it was recently shown that accurate nonlinear models could be identified directly from lift and drag measurements on the cylinder [91].

We generate data by direct numerical simulation of the two-dimensional Navier-Stokes equations using the immersed boundary projection method [156, 33], resulting in 151 snapshots in time with spatial resolution of 199×449. As in [18], we extract the time

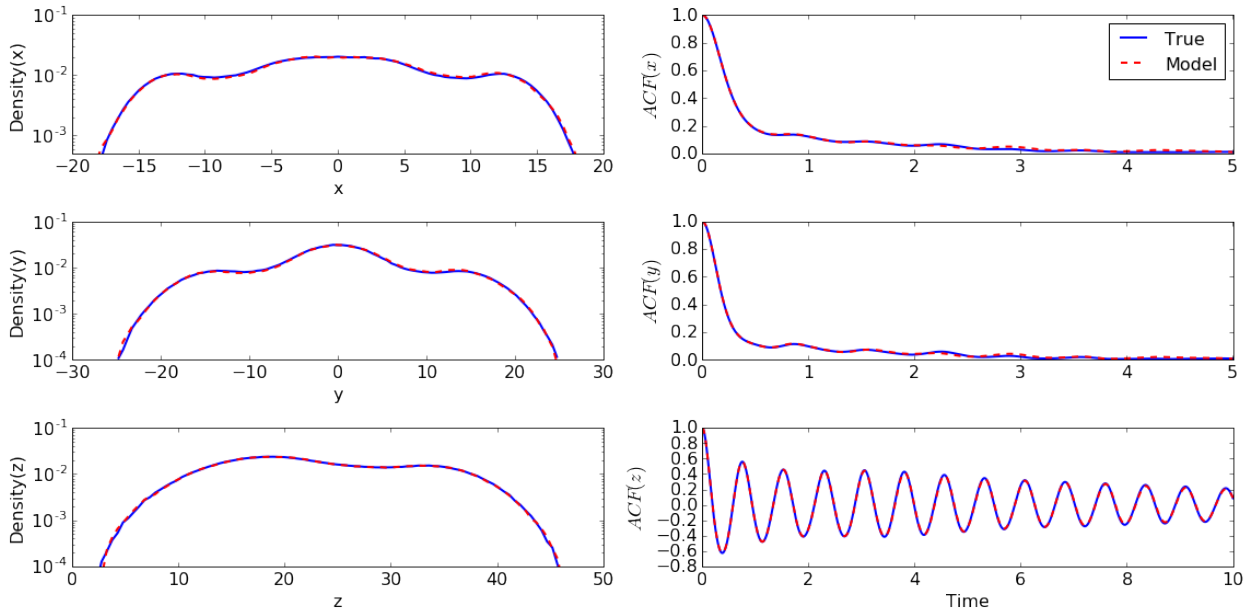


Figure 3.9: Marginal phase space densities and autocorrelation function for Lorenz system.

series of the first two proper orthogonal decomposition (POD) modes and the shift mode of Noack *et al.* [115] as our clean training data; these modes are shown in Fig. 3.10. We add noise to the data following projection onto the low-dimensional subspace. The mean ℓ^2 errors for the measurement noise approximation are $1.019\text{e}{-4}$ and 0.0504 for the cases of 0% and 1% noise, respectively. We do not compute error metrics for vector field accuracy since the true vector field is unknown. However, the qualitative behavior of observations and model predictions match the training data, shown in Fig. 3.11.

3.3.5 Double pendulum

In all of the example investigated so far, the true equations of motion have been simple polynomials in the state, and would therefore be easily represented with a sparse regression method such as the sparse identification of nonlinear dynamics (SINDy) [18]. The utility of a neural network for approximating the vector field becomes more clear when

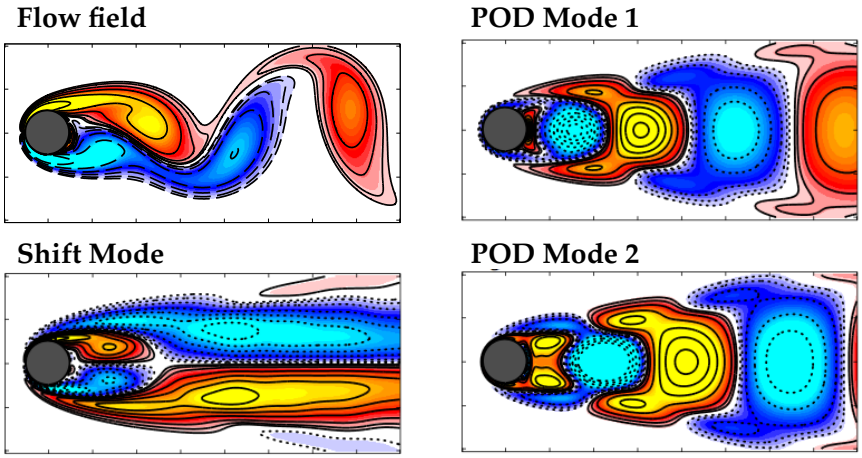


Figure 3.10: Modes from simulation of Navier-Stokes equation. The x, y, z coordinates used in this test problem are time series obtained by projecting the full flow field onto POD mode 1, POD mode 2, and the shift mode, respectively.

we consider dynamics that are not easily represented by a standard library of elementary functions. The double pendulum is a classic mechanical system exhibiting chaos, with dynamics that would are challenging for a library method, although previous work has recovered the Hamiltonian via genetic programming [151]. The double pendulum may be modeled by the following equations of motion in terms of the two angles θ_1 and θ_2 of the respective pendula from the vertical axis and their conjugate momenta p_1 and p_2:

$$\begin{aligned}
\dot{\theta}_1 &= \frac{l_2 p_1 - l_1 p_2 \cos(\theta_1 - \theta_2)}{l_1^2 l_2 \left(m_1 + m_2 \sin^2(\theta_1 - \theta_2)\right)} \\
\dot{\theta}_2 &= \frac{-m_2 l_2 p_1 \cos(\theta_1 - \theta_2) + (m_1 + m_2) l_1 p_2}{m_2 l_1 l_2^2 \left(m_1 + m_2 \sin^2(\theta_1 - \theta_2)\right)} \\
\dot{p}_1 &= -(m_1 + m_2) g l_1 \sin(\theta_1) - C_1 + C_2 \sin(2(\theta_1 - \theta_2)) \\
\dot{p}_2 &= -m_2 g l_2 \sin(\theta_2) + C_1 - C_2 \sin(2(\theta_1 - \theta_2)),
\end{aligned} \tag{3.37}$$

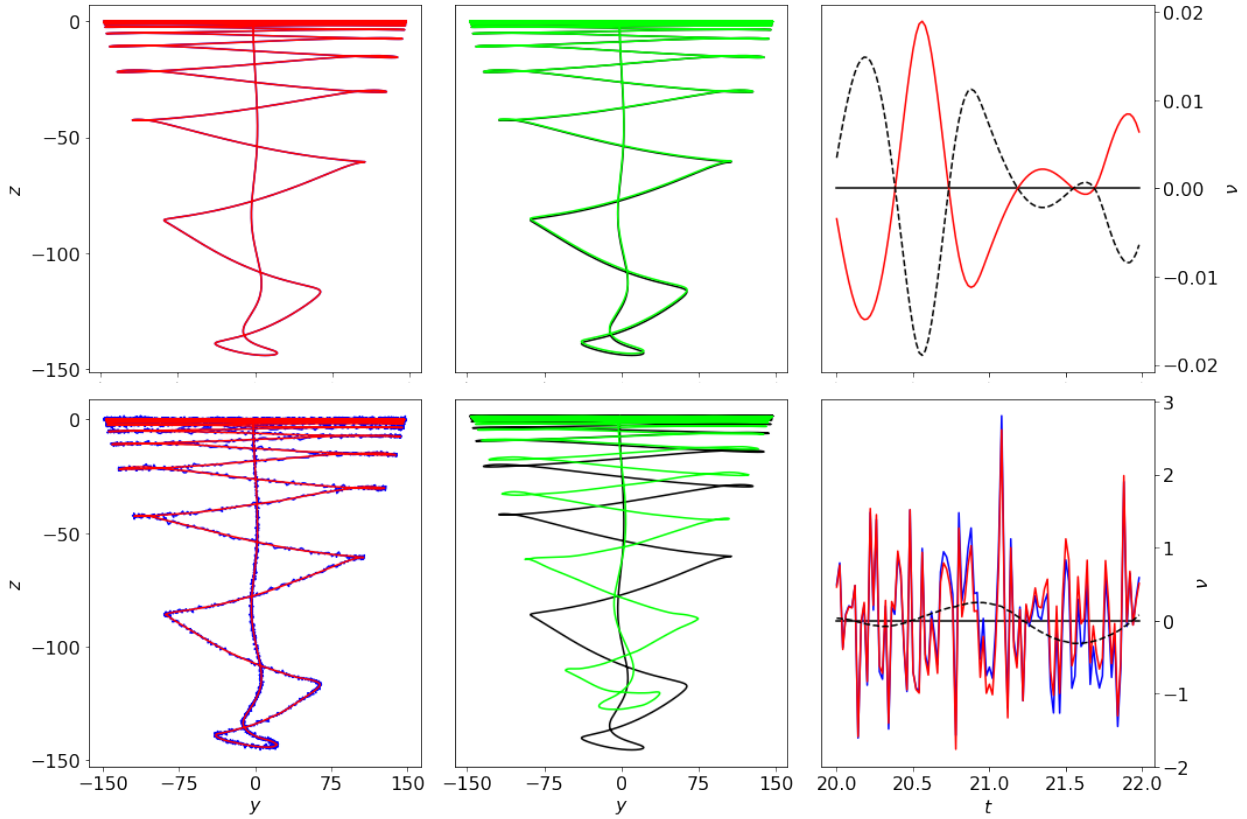


Figure 3.11: Results for reduced basis of flow around a cylinder with increasing magnitudes of measurement noise. Top row: Clean data. Bottom row: 1% Gaussian noise. Left column: Observations Y in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. Note green trajectory has been extended to T_{max} five times that of training data. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in estimation of measurement noise as dashed black line.

where

$$C_1 = \frac{p_1 p_2 \sin(\theta_1 - \theta_2)}{l_1 l_2 (m_1 + m_2 \sin^2(\theta_1 - \theta_2))}$$
$$C_2 = \frac{m_2 l_2^2 p_1^2 + (m_1 + m_2) l_1^2 p_2^2 - 2 m_2 l_1 l_2 p_1 p_2 \cos(\theta_1 - \theta_2)}{2 l_1^2 l_2^2 (m_1 + m_2 \sin^2(\theta_1 - \theta_2))^2}, \tag{3.38}$$

and $l_1 = l_2 = 1$ are the lengths of the upper and lower arms of the pendulum, $m_1 = m_2 = 1$ the respective point masses, and $g = 10$ is the acceleration due to gravity. Numerical solutions to (3.37) are obtained using a symplectic integrator starting from the initial condition $(\theta_1, \theta_2, p_1, p_2) = (1, 0, 0, 0)$ from $t = 0$ to $t = 50$ with a timestep of $\Delta t = 0.01$. A symplectic or variational integrator is required to ensure that energy is conserved along the trajectory [180, 106]. It is important to note that this initial condition represents a low-energy trajectory of the double pendulum with non-chaotic dynamics existing on a bounded region of phase space. Neither pendulum arm makes a full revolution over the vertical axis. For higher energies, the method presented in this thesis did not yield satisfying results.

We construct a data-driven approximation to the vector field f in (3.37) using a neural network for the vector field with five hidden layers, each containing 64 nodes, embedded in a four-step Runge-Kutta scheme to approximate the discrete-time flow map F. Artificial measurement noise is added to the trajectory with magnitudes up to 10%. Examples of training data, noise estimates, and numerical solutions to the learned dynamics are shown in Fig. 3.12 for noise levels of 0, 1, and 10 percent of the standard deviation of the dataset. Summary error measures for the learned dynamics and measurement noise of the double pendulum are shown in table 3.5. In all cases, it can be seen that the error is effectively separated from the training data, resulting in accurate model predictions, even for relatively large noise magnitudes.

Table 3.5: Error for double pendulum with varying noise.

% Noise	0	1	5	10
E_N	$5.823e-7$	$5.835e-5$	$1.200e-3$	$3.399e-3$
E_f	$5.951e-3$	$6.192e-3$	$8.575e-3$	$1.444e-2$

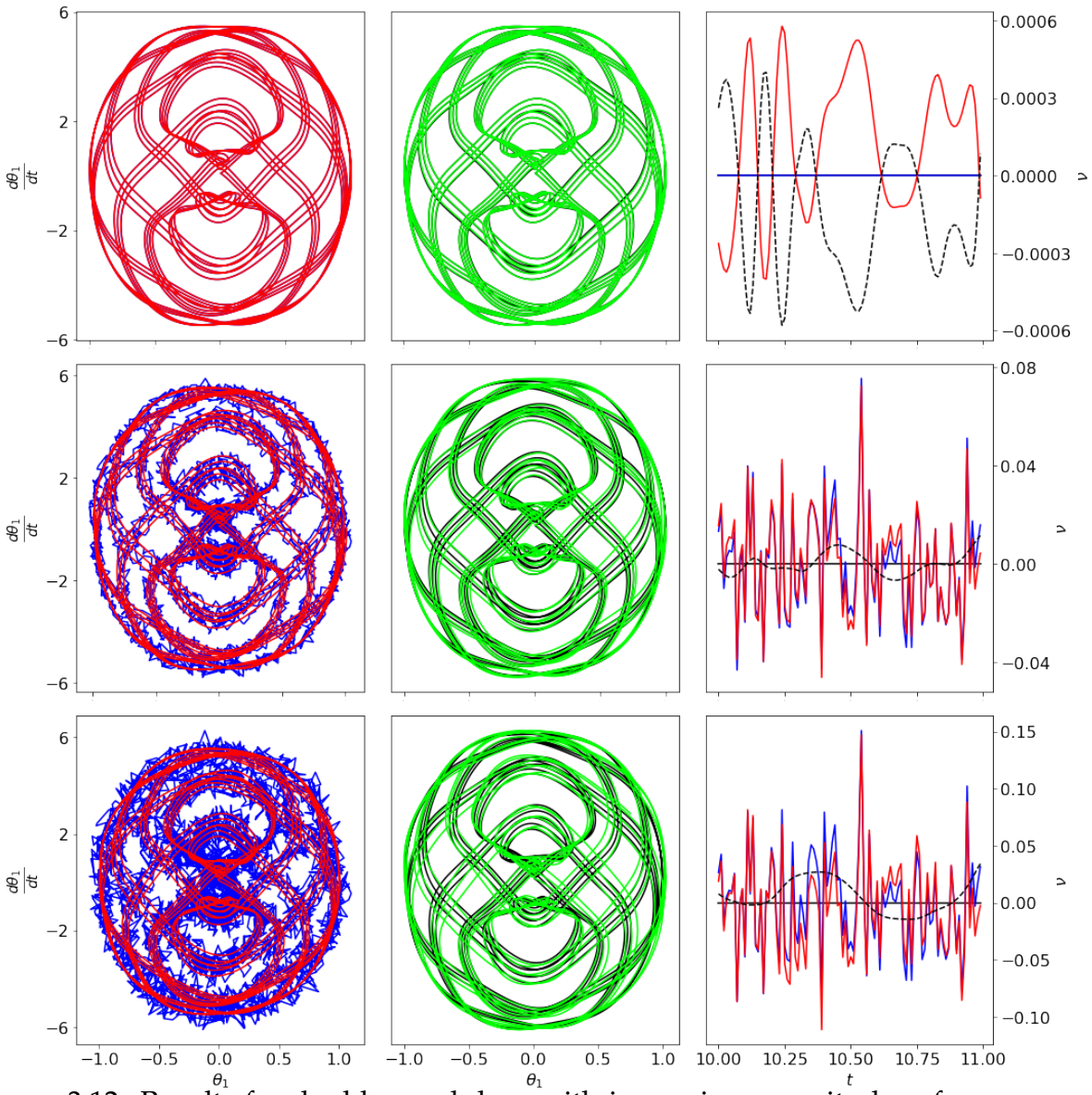


Figure 3.12: Results for double pendulum with increasing magnitudes of measurement noise. Top row: Clean data. Second row: 5% Gaussian noise. Bottom row: 10% Gaussian noise. Left column: Observations $\mathbf{Y}$ in blue and learned state $\mathbf{Y} - \hat{\mathbf{N}}$ in red. Middle column: True state $\mathbf{X}$ in black and forward orbit of $\mathbf{x}_1$ under $\hat{F}_{\boldsymbol{\theta}}$ in green. Right column: True measurement noise $\mathbf{N}$ in blue, learned measurement noise $\hat{\mathbf{N}}$ in red, and error in estimation of measurement noise as dashed black line.

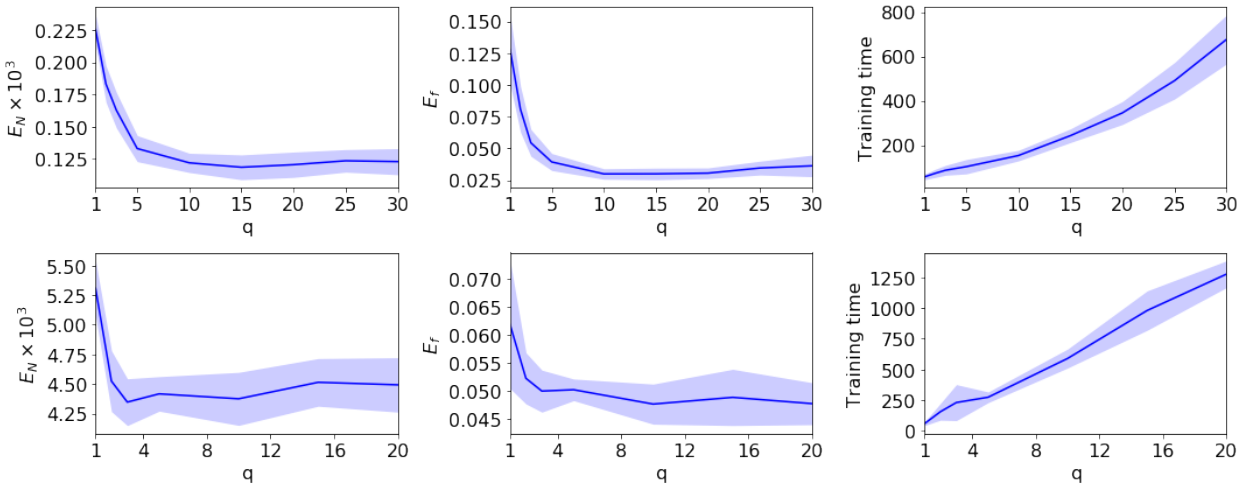


Figure 3.13: Error metrics and training time for varying q. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36).

3.3.6 *Sensitivity of results to parameters*

Equation (3.18) involves four hyperparameters: q, γ, β, and the decay rate of ω_i. In this section we consider the sensitivity of various errors with respect to changes in these parameters. In keeping with the notation from Sec. 3.2.2, we let $\omega_i = \omega_0 \rho^{-|i|}$ with $\omega_1 = 1$ so that the decay in ω_i is geometric with constant ρ^{-1}. We test each parameter both for the cubic oscillator and Lorenz system. Since the Lorenz system is chaotic, a perfect representation of f will quickly accrue error due to differences in numerical schemes, so E_F is not a good measure of accuracy. We define tracking time as,

$$\text{Tracking time} = dt \cdot \min\{j : \|\hat{F}^j_{\boldsymbol{\theta}}(\mathbf{x}_1) - \mathbf{x}_{j+1}\|_2 > 5\}, \tag{3.39}$$

and report this metric in place of E_F where applicable. Shaded regions on each figure correspond to one standard deviation. Training time, where reported is given as wall time for the optimizer to achieve a given threshold in relative change of the cost function over one iteration.

Figure 3.13 shows the error and training time for values of q ranging from 1 to 20

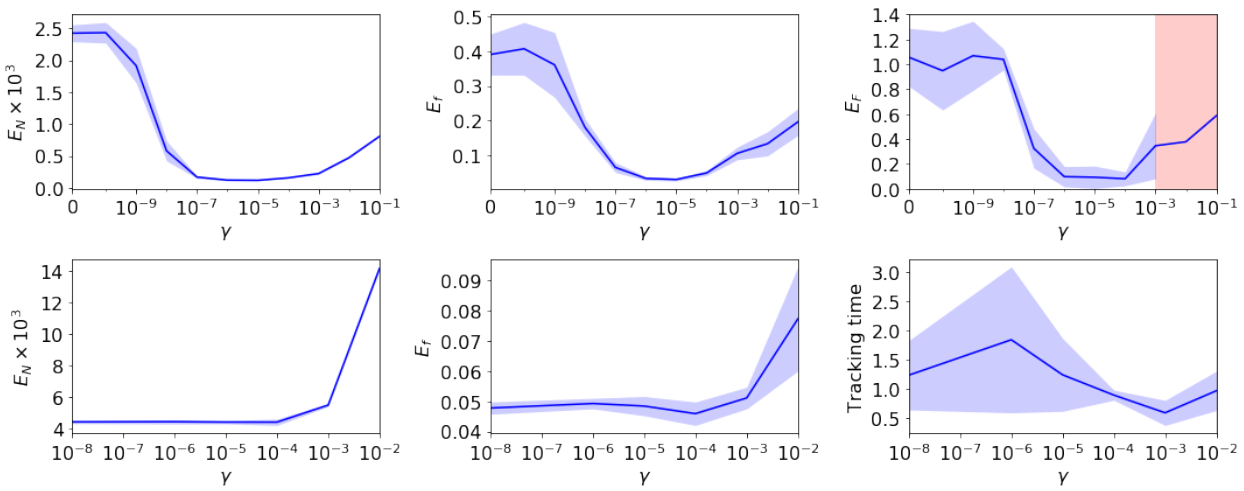


Figure 3.14: Error metrics and training time for varying regularization γ on measurement noise estimates. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36).

with other parameters fixed at $\gamma = 10^{-5}$, $\beta = 10^{-8}$ and $\rho^{-1} = 0.9$. Ten random noise instantiations are tested for each parameter regime.

Figure 3.14 shows the error and training time for values of γ ranging from 0 to 0.1 for the cubic oscillator and from 10^{-8} to 10^{-2} for the Lorenz system where non-zero values are equally spaced on a log scale. A subset of the trials for the cubic oscillator model with $\gamma > 10^{-3}$ diverged, yielding non-numeric E_F. Therefore, standard deviation has been omitted and the median, rather than mean, is plotted. We have also omitted the unregularized case for Lorenz due to excessive computational cost. Other parameters are fixed at $q = 10$, $\beta = 10^{-8}$ and $\rho^{-1} = 0.9$. Ten random noise instantiations are tested for each parameter regime for the cubic oscillator model and five for the Lorenz system.

Figure 3.15 shows the error and training time for values of β ranging from 0 to 0.01 where non-zero values are equally spaced on a log scale. Other parameters are fixed at $q = 10$, $\gamma = 10^{-5}$ and $\rho^{-1} = 0.9$. Five random noise instantiations are tested for each parameter regime. We note that below a threshold of roughly 10^6, the exact value of β had little effect. Setting $\beta = 0$ slightly increased the error variance for the cubic oscillator

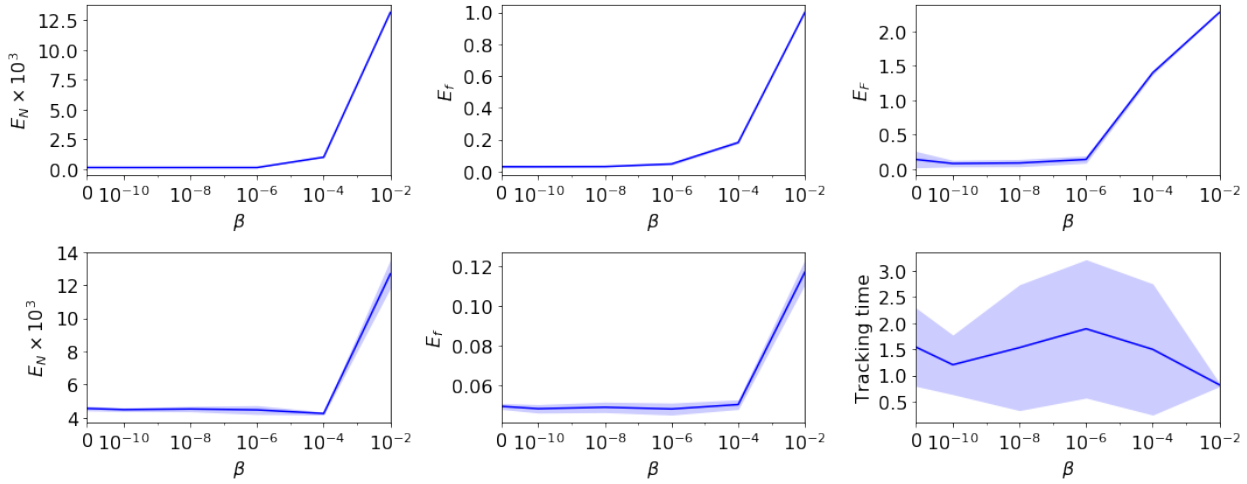


Figure 3.15: Error metrics and training time for varying regularization β on neural network weights. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36).

model while having the opposite effect for the Lorenz system. Variation in β did not significantly change average training time for either system. Higher γ resulted in slightly faster training for the Lorenz system while not having significant effect on training time for the cubic oscillator model.

Figure 3.16 shows the error and training time for values of ρ^{-1} ranging from 0.1 to 1. Other parameters are fixed at $q = 10$, $\gamma = 10^{-5}$ and $\beta = 10^{-8}$. Five random noise instantiations are tested for each parameter regime. Some test trajectories for the cubic oscillator when $\rho^{-1} = 0.1$ diverged, resulting in mean $E_F \sim \mathcal{O}(10^{16})$, so we have omitted this data in the figure.

3.4 Cautionary remarks on neural networks and overfitting

Neural networks are fundamentally interpolation methods [102] with high-dimensional parameter spaces that allow them to represent arbitrarily complex functions [62]. The large number of free parameters required for arbitrary function fitting also creates the risk of overfitting, necessitating significant volumes of rich training data. Many successful applications of neural networks employ regularization techniques such as ℓ^2 regular-

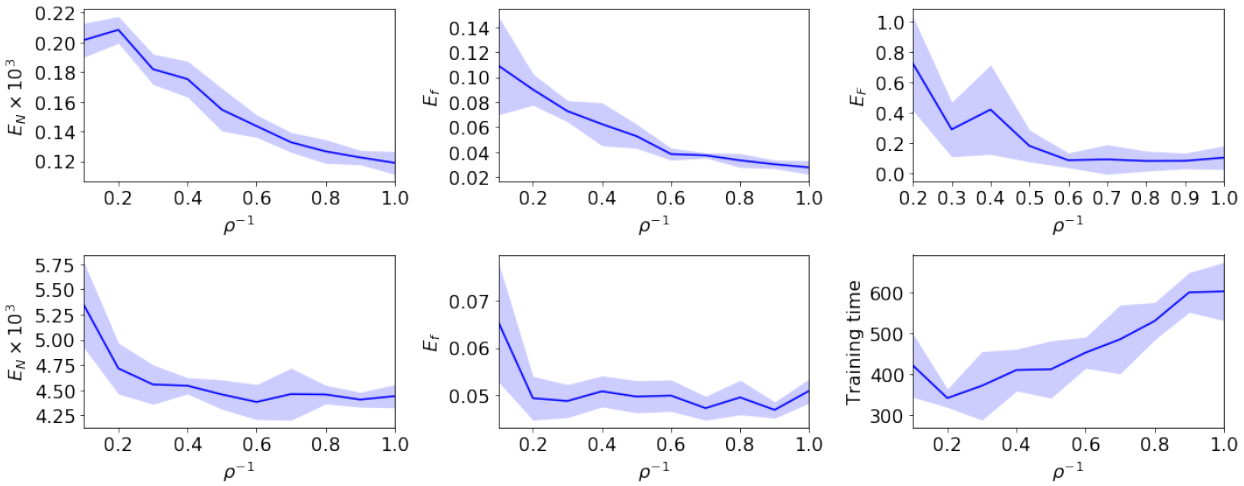


Figure 3.16: Error metrics and training time for varying geometric decay constant ρ^{-1} for multistep prediction loss. Top row: Cubic oscillator model (3.35). Bottom row: Lorenz system (3.36).

ization, dropout [153], and early stopping [23] to help prevent overfitting. In computer vision, data augmentation through preprocessing, such as random rotations and image flipping, helps prevent overfitting and allows single labeled examples of training data to be reused without redundancy. Recent innovations in the use of neural networks for dynamical systems forecasting have included regularization of the network Jacobian [118], but data augmentation does not have a clear analog in dynamical systems. This section will explore key limitations and highlight pitfalls of training neural networks for dynamical systems.

Section 3.3 demonstrated the ability of our proposed method to accurately represent dynamics from limited and noisy time-series data. In particular, the dynamics shown in the Lorenz equation, fluid flow, and double pendulum examples all evolved on an attractor, which was densely sampled in the training data. In each case, trajectories integrated along the learned dynamics remain on the attractor for long times. This indicates that our neural network is faithfully interpolating the vector field near the attractor. Fortunately, real-world data will often be sampled from an attractor, providing sufficiently rich data

to train a model that is valid nearby. However, care must be taken when extrapolating to new initial conditions or when making claims about the vector field based on models learned near the attractor. In particular, transient data from off of the attractor may be essential to train models that are robust to perturbations or that are effective for control, whereby the attractor is likely modified by actuation [17].

By analyzing a known dynamical system such as the Lorenz equations, we are able to quantify the performance of the method in approximating the vector field when given only data on the attractor, or when given many trajectories containing transients. To do so, we extend our previous method to fit multiple datasets of observations to the same dynamical system. Given a set of p trajectories, $\{\mathbf{Y}_k\}_{k=1}^p$, all having the same underlying dynamics, we adapt the cost function given by (3.18) to

$$\mathcal{L}_{\text{multi}}\left(\boldsymbol{\theta}, \{\hat{\mathbf{N}}_k\}_{k=1}^p, \{\mathbf{Y}_k\}_{k=1}^p\right) = \sum_{k=1}^{p} \mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{N}}_k, \mathbf{Y}_k). \tag{3.40}$$

Based on (3.40), we compare the accuracy of the learned vector field from datasets of comparable size, obtained either from a single trajectory or from many short trajectories. Figure 3.17 shows the data and learned vector field on the plane $z = 25$ for the Lorenz system trained either from a single trajectory of length $m = 10000$, or from 50 individual trajectories of length 200, each from a random initial condition off the attractor. The exact vector field is shown for comparison. Unsurprisingly, the long trajectory does not result in accurate interpolation of the vector field off the attractor, while training from many trajectories with transients results in a more accurate model. We note that despite the apparent inaccuracies of the vector field displayed in the bottom left panel, the same dataset was used to construct the forecasting model analyzed in Fig. 3.9, where long term statistics are shown to be accurate.

Neural networks with different parameterizations will also result in varying behavior away from training data. To illustrate this, we consider three parameterizations of f for the fluid flow dataset, using 3 hidden layers of size 64 or 128, as well as 5 hidden layers

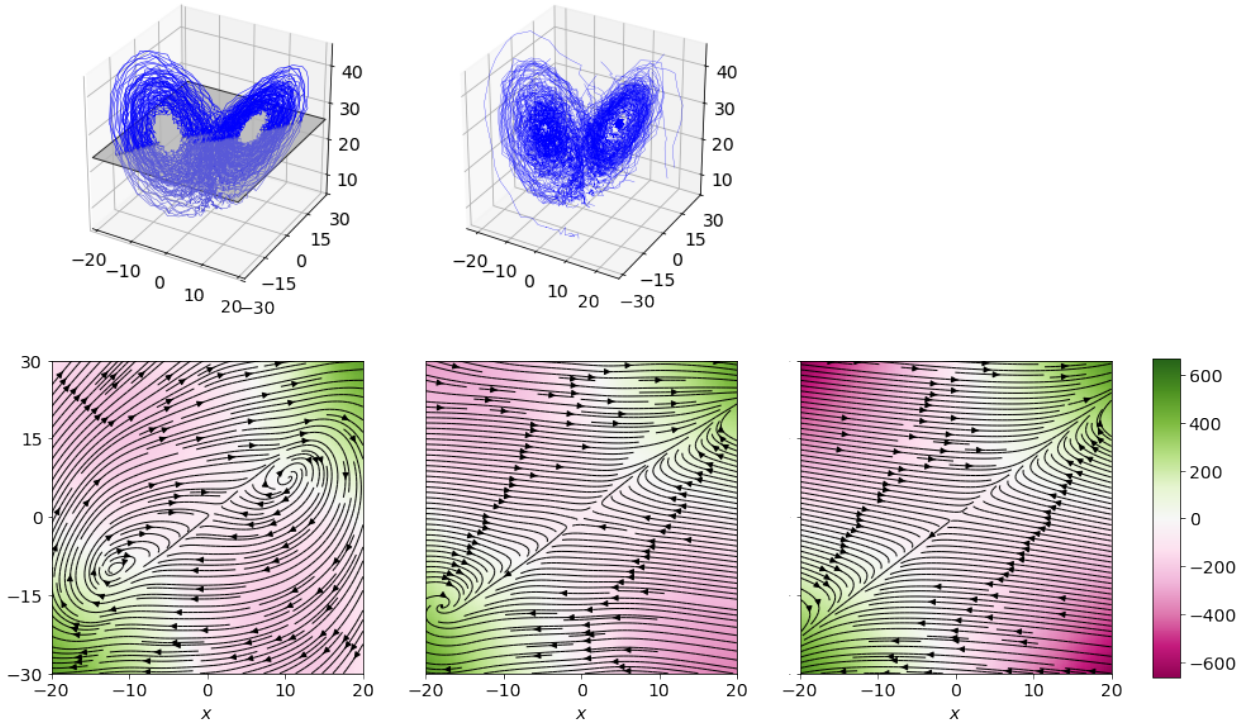


Figure 3.17: Learned vector fields for the Lorenz system with 5% noise. Left: Single trajectory (top) and learned vector field (bottom) at Poincare section $z = 25$. xy components of $\hat{f}_{\boldsymbol{\theta}}$ shown in stream plot with z component given by color. Center: 50 short trajectories and learned vector field using (3.40). Right: True vector field.

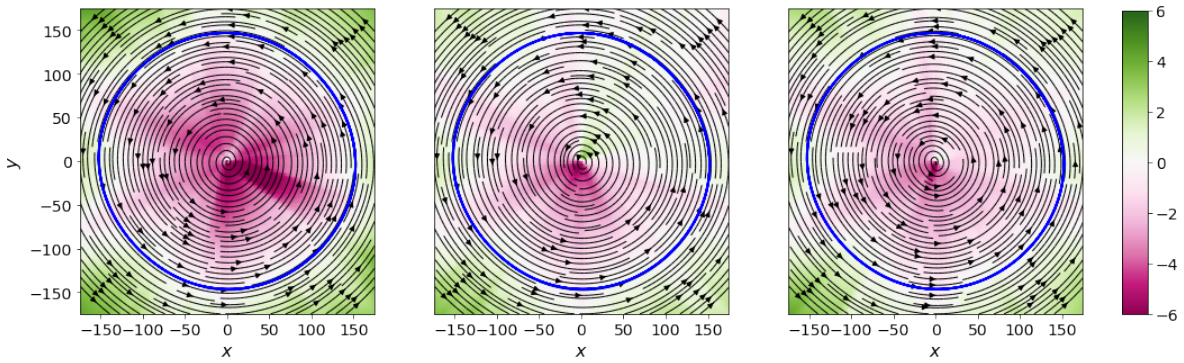


Figure 3.18: Learned vector fields for the reduced order system from flow around a cylinder with different sizes of neural network. Left: 3 rows of 64 nodes each. Center: 3 rows of 128 nodes. Right: 5 rows of 64 nodes. Mean field model exhibits radial symmetry.

of size 64. The resulting vector fields along the $z = 0$ plane are shown in Fig. 3.18. The limit cycle in the training data is shown in blue to indicate regions in the domain near training data. Each of the three parameterizations accurately models the forward orbit of the initial condition from the training data and converges to the correct limit cycle. However, all networks fail to identify a fundamental radial symmetry present in the problem, indicating that a single test trajectory is insufficient to enable forecasting for general time series. A similar phenomena is observed for the linear problem in Fig. 3.3. For data on the attractor, though, these specific parameterizations may be sufficient.

3.5 Discussion

In this work we have presented a machine learning technique capable of representing the vector field of a continuous dynamical system from limited, noisy measurements, possibly with irregular time sampling. By explicitly considering noise in the optimization, we are able to construct accurate forecasting models from datasets corrupted by considerable levels of noise and to separate the measurement error from the underlying state. Our methodology constructs discrete time-steppers using a neural network to represent the underlying vector field, embedded in a classical Runge-Kutta scheme, enabling seamless learning from datasets unevenly spaced in time. The constrained learning architecture exploits structure in order to provide a robust model discovery and forecasting mathemat-

ical framework. Indeed, by constraining learning, the model discovery effort is focused on discovering the dynamics unencumbered by noise and corrupt measurements.

Using a neural network to interpolate the underlying vector field of an unknown dynamical system enables flexible learning of dynamics without any prior assumptions on the form of the vector field. This is in contrast to library-based methods, which require the vector field to lie in the span of a pre-determined set of basis functions. Neural networks are substantially more computationally expensive to train than convex methods, and they are more prone to error when evaluated away from training data, as demonstrated in Sec. 3.4. It follows that the proposed method is likely most useful for studying systems evolving on a bounded attractor and where computational time needed for model training is not of foremost concern.

The combination of neural networks and numerical time-stepping schemes suggests a number of high-priority research directions in system identification and data-driven forecasting. Future extensions of this work include considering systems with process noise, a more rigorous analysis of the specific method for interpolating f, including time delay coordinates to accommodate latent variables [16], and generalizing the method to identify partial differential equations. Rapid advances in hardware and the ease of writing software for deep learning will enable these innovations through fast turnover in developing and testing methods.

Chapter 4

SMOOTHING BY SOFT ADHERENCE TO GOVERNING EQUATIONS

This chapter develops a framework for smoothing noisy time series data where governing equations are known up to parameters. The chapter is organized as follows: In Sec. 4.2 we formalize the problem statement, provide an overview of the mathematical justification for our methods, and introduce a novel computational method for smoothing data based on soft adherence to a time-stepping scheme. In Sec. 4.3 we provide numerical results for a few standard test problems in data assimilation and draw comparisons to the Ensemble Rauch-Tung-Streibel Smoother. In addition to Gaussian measurement error, error from Ornstein-Uhlenbeck processes, heavy tailed noise, and Gaussian error with non-zero mean are also considered. Section 4.4 contains a discussion of the method and remarks for further work.

4.1 Introduction

Experimental measurement noise is ubiquitous across all fields of science. Such noise may undermine data-driven methods such as proper orthogonal decomposition [60], dynamic mode decomposition [85], forecasting, and parameter estimation. Consequently, many methods have been developed for state estimation and smoothing.

Perhaps the most commonly used state estimation algorithm is the Kalman filter [77]. Kalman filters sequentially estimate the state of a linear stochastic process by tracking mean and covariance estimates. Under linear dynamics, this process may be carried out exactly since Gaussian distributions are closed under affine transformations. The Kalman filter scales linearly in the length of the time series and may be computed on line. These

computational properties allowed it to see a variety of applications in engineering systems.

There are several methods for generalizing the Kalman filter to nonlinear dynamical systems. The extended Kalman filter (EKF) [107] computes the propagation of error covariance through the dynamics using a linearization at each timestep. The EKF has not been widely used due to several difficulties. Computational speed may be significantly decreased, in particular for high dimensional systems, due to needing a Jacobian at each timestep and first order accuracy may not be sufficient o accurately capture changes in covariance. Several ensemble based methods exist which improve on the EKF for filtering nonlinear systems. The Unscented Kalman Filter [75] samples points deterministically from the state estimate distribution, passes them through the dynamics, and finds new covariance based on the distribution. Alternatively the Ensemble Kalman Filter (EnKF) [43] samples points randomly according to the state estimate and covariance. The EnKF exhibits surprisingly good performance on high dimensional problems and, with some modifications such as covariance inflation [63] may be considered state of the art. However, for highly nonlinear problems, assuming Gaussian error statistics may be problematic [100]. Particle filters were introduced to address concerns that Gaussian error statistics are not maintained when dynamics are nonlinear [55] by allowing for arbitrary distributions representing state density. However, they have been shown to have difficulty with high dimensional systems [6].

The Kalman and Rauch-Tung-Striebel extend on the Kalman filter to include a backwards pass over the data, conditioning state estimates on data over the full interval collected, rather than only prior measurements. The success of these methods is contingent on some degree of accuracy for the forward pass of the Kalman filter. For chaotic or highly nonlinear dynamics, this may be difficult. The problem is further complicated if governing equations are not fully known, in which case it is likely that state estimation methods must be alternated with system identification.

This chapter presents a novel method for smoothing experimental data where dynam-

ics are known up to a set of coefficients. The proposed method acts by exploiting the deterministic nature of the known underlying governing equations. Specifically, we formulate an optimization whereby we enforce adherence to a time-stepping constraint for the deterministic system, thus separating the noise from the dynamics. We emphasize that the work does not share the generality or rigor of Bayesian sequential data assimilation methods such as the ensemble Kalman filter but does yield more accurate predictions on problems with deterministic dynamics. Specifically, it optimizes over full time trajectories. It is also possible to perform parameter estimation with the algorithm presented in this work without alternating between state and parameter estimation. Finally, the method presented here is robust to non-Gaussian noise, stiff and highly nonlinear dynamics, and even noise with non-zero mean. We believe this work forms a meaningful contribution to any field where noisy measurements of smooth dynamics systems must be considered.

4.2 Methods

This work leverages mathematical optimization to find state estimates for a measured time series by enforcing soft adherence to known dynamics. We use known time-stepping architectures to construct measures of accuracy for estimated time series to our governing equations. We also enforce the state estimate to lie close to the measured data. The proposed method is able to estimate the true state to high precision even in the case of temporally autocorrelated noise or noise with non-zero mean. Section 4.2.1 provides a brief overview of the problems considered in this work and Sec. 4.2.2 introduces the proposed computational framework for denoising as well as a discussion of initialization and optimization.

4.2.1 *Problem formulation*

We consider observations $\mathbf{y}_j$ of a continuous process $\mathbf{x}$ given as,

$$\mathbf{y}_j = \mathbf{x}_j + \boldsymbol{\nu}_j, \tag{4.1}$$

where $\mathbf{x}_j \in \mathbb{R}^n$ denotes the value of $\mathbf{x}$ at time t_j for $j = 1, \ldots, m$ and ϵ_j is measurement error. We further presume that $\mathbf{x}$ obeys the autonomous differential equation,

$$\dot{\mathbf{x}} = f(\mathbf{x}), \tag{4.2}$$

where $\mathbf{x}$ is the state and $f(\mathbf{x})$ the velocity. Many modern techniques in data-assimilation and smoothing use sequential Bayesian filters that make forecast estimates of the state $\mathbf{x}_{j+1}$ from predictions of $\mathbf{x}_j$ and then assimilate that forecast using estimates of the error covariance and the measurement $\mathbf{y}_{j+1}$. In contrast, this work seeks to estimate states $\mathbf{x}_j$ for each time step by balancing two criteria; 1) the estimate states should fit the prescribed dynamics and 2) the estimated state must be as close as possible to the measurements.

4.2.2 *Computational methods*

The general form of a Runge-Kutta scheme [84] is,

$$\mathbf{x}_{j+1} = \mathbf{x}_j + h_j \sum_{i=1}^{s} b_i \mathbf{k}_j^{(i)} \tag{4.3}$$

$$\mathbf{k}_j^{(i)} = f\left(\mathbf{x_j} + h_j \sum_{l=1}^{s} a_{il} \mathbf{k}_j^{(l)},\, t_j + c_i h_j\right), \tag{4.4}$$

where matrix $\mathbf{A} \in \mathbb{R}^{s \times s}$, and vectors $\mathbf{b}, \mathbf{c} \in \mathbb{R}^s$ define specific weights used in the timestepper. Given $\mathbf{x}_j$ the solution to the system of equations in (4.3) (4.4) gives the state $\mathbf{x}_{j+1}$ at the subsequent timestep and values of f at intermediate stages. If $a_{ij} = 0$ for $j \geq i$ then the method is explicit and there is no need to solve a set of algebraic equations, but these

methods are generally less reliable for stiff problems. For dense $\mathbf{A}$ it is possible to obtain $\mathcal{O}(h^{2s})$ accurate time stepping schemes. For the remainder of this text, time dependence in f will be omitted from notation, though including it will not change any analysis.

We note that for any pair of states $(\mathbf{x}_j, \mathbf{x}_{j+1})$ from the ground truth solution for the denoising problem there exists a set of intermediate values of the velocity $\{\mathbf{k}_j^{(i)}\}_{i=1}^s$ such that equations (4.3) and (4.4) are satisfied to within the order of accuracy for the timestepper being considered. This observation forms the foundation for the methods considered in this work. For any pair of state estimates $(\mathbf{x}_j, \mathbf{x}_{j+1})$ and intermediate values of the derivative $\{\mathbf{k}_j^i\}_{i=1}^s$ we define cost functions that measure the ability of the intermediate values of the derivative to predict the next timestep, and the consistency of the intermediate values to their definition in (4.3) (4.4). These are given by,

$$L_1^{(j)}(\mathbf{X}, \mathbf{K}) = \left\| \mathbf{x}_{j+1} - \left(\mathbf{x}_j + h_j \sum_{i=1}^{s} b_i \mathbf{k}_j^{(i)} \right) \right\|_2^2 \tag{4.5}$$

$$L_2^{(j,i)}(\mathbf{X}, \mathbf{K}) = \left\| \mathbf{k}_j^{(i)} - f\left(\mathbf{x}_j + h_j \sum_{l=1}^{s} a_{il} \mathbf{k}_j^{(l)} \right) \right\|_2^2, \tag{4.6}$$

where $\mathbf{X} = \{\mathbf{x}_j\}_{j=1}^m$ and $\mathbf{K} = \{(\mathbf{k}_j^{(1)}, \ldots, \mathbf{k}_j^{(s)})\}_{j=1}^m$. For a time series of state estimates and estimates of intermediate values of the derivative, we define a global cost function evaluating the fidelity of the time series to the prescribed dynamics by summing equations (4.5) and (4.6) over each timestep. Since equations (4.5) and (4.6) do not reference the measurements $\mathbf{Y}$, we include a measure of closeness to the measured data as $g(\mathbf{X} - \mathbf{Y})$. The cost function balancing accuracy to the timestepping scheme with closeness to data is given by,

$$L(\mathbf{X}, \mathbf{K}) = \sum_{j=1}^{m-1} \left(L_1^{(j)}(\mathbf{X}, \mathbf{K}) + \sum_{i=1}^{s} L_2^{(j,i)}(\mathbf{X}, \mathbf{K}) \right) + g(\mathbf{X} - \mathbf{Y}), \tag{4.7}$$

where a suitable choice for $g(\cdot)$ may be the commonly used 2-norm $\lambda\|\cdot\|_2^2$ for Gaussian noise or the more robust 1-norm $\lambda\|\cdot\|_1$ for handling heavy tailed noise.

Optimization over (4.7) yields state estimates with considerable error. Terms in (4.7) from the sum over (4.6) may be considerable different in magnitude than those from (4.6). Scaling the time variable by a suitable value of α so that $\tau = \alpha t$ and velocity is effectively scaled by α^{-1} provides a remedy for the imbalance but this is not practice in an application setting. The appropriate scaling factor may be unclear since the magnitude of f evaluated on the measurements may be significantly different than when evaluated on the optimal state estimate. A more reliable fix is to optimize over intermediate values of the state rather than of the velocity. For f with non-singular Jacobian at each $\mathbf{x_j} + h_j \sum_{l=1}^{s} a_{il}\mathbf{k}_j^{(l)}$ the implicit function theorem tells us we can find an equivalent expression to (4.3),(4.4) involving only values of $\mathbf{x}$. Inverting f at each $\mathbf{k}_j^{(i)}$ in eq. (4.4) gives,

$$\mathbf{x}_j^{(i)} = f^{-1}(\mathbf{k}_j^{(i)}) = \mathbf{x_j} + h_j \sum_{l=1}^{s} a_{il}\mathbf{k}_j^{(l)}. \tag{4.8}$$

Replacing each $\mathbf{k}_j^{(i)}$ in equations (4.3) and (4.4) with $f(\mathbf{x}_j^{(i)})$ and applying f^{-1} to both sides of (4.4) we obtain a Runge-Kutta scheme where intermediate values are taken in the domain of f rather than the range.

$$\mathbf{x}_{j+1} = \mathbf{x}_j + h_j \sum_{i=1}^{s} b_i f\left(\mathbf{x}_j^{(i)}\right) \tag{4.9}$$

$$\mathbf{x}_j^{(i)} = \mathbf{x_j} + h_j \sum_{l=1}^{s} a_{il} f\left(\mathbf{x}_j^{(l)}\right). \tag{4.10}$$

Using equations (4.9) and (4.10) we can follow the same process as before to construct a cost function over the set of variables $\mathbf{X} = \{\mathbf{x}_j\}_{j=1}^m$ and $\tilde{\mathbf{X}} = \{(\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(s)})\}_{j=1}^m$. The

resulting function for Runge-Kutta denoising is given by,

$$L_{RKD}(\mathbf{X}, \tilde{\mathbf{X}}) = \sum_{j=1}^{m-1} \left(\mathcal{L}_1^{(j)} \left(\mathbf{X}, \tilde{\mathbf{X}} \right) + \sum_{i=1}^{s} \mathcal{L}_2^{(j,i)} \left(\mathbf{X}, \tilde{\mathbf{X}} \right) \right) + g(\mathbf{X} - \mathbf{Y}), \tag{4.11}$$

where,

$$\mathcal{L}_1^{(j)} \left(\mathbf{X}, \tilde{\mathbf{X}} \right) = \left\| \mathbf{x}_{j+1} - \left(\mathbf{x}_j + h_j \sum_{i=1}^{s} b_i f \left(\mathbf{x}_j^{(i)} \right) \right) \right\|_2^2 \tag{4.12}$$

$$\mathcal{L}_2^{(j,i)} \left(\mathbf{X}, \tilde{\mathbf{X}} \right) = \left\| \mathbf{x}_j^{(i)} - \left(\mathbf{x}_j + h_j \sum_{l=1}^{s} a_{il} f \left(\mathbf{x}_j^{(l)} \right) \right) \right\|_2^2. \tag{4.13}$$

Note that (4.11) is very similar to (4.7) but where we are now optimizing over intermediate values of the state rather than the velocity. A more detailed comparison between the two is given is Sec. 4.3.4. It is also possible to include additional terms in (4.11) to penalize properties of the state estimate such as variation in derivatives to enforce smoothness.

Many state of the art algorithms for system identification from noisy data employ an expectation-maximization framework to alternate between smoothing and parameter estimation [50]. In contrast, the method considered in this work extends trivially to allow for learning state estimates and model parameters in a single minimization problem. Where f is known up to a set of parameters θ we replace f by $\hat{f}_\theta$ in (4.12)-(4.13) and optimize over the state estimate with intermediate values and model parameters.

We minimize (4.11) using the quasi-Newton solver L-BFGS [183]. Derivatives of the cost function are evaluated using automatic-differentiation software implemented in the Tensorflow library in Python [1]. Initial estimates for the states are obtained by applying a naive smoother to the measured data and intermediate state estimates $\mathbf{x}_j^{(i)}$ are obtained by a linear interpolation between initial estimates of $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$.

4.3 Results

In this section we present several numerical examples of increasing complexity and dimension. We test the denoising algorithm on the Lorenz 63 (simply called the Lorenz system in previous chapters), Lorenz 96, Kuramoto-Sivashinsky, and nonlinear Schrödinger equations. We also provide examples of the denoising algorithm in cases where model parameters are unknown. In each case the method is tested with white noise added to the state, but more complicated cases including noise drawn from an Ornstein-Uhlenbeck process, heavy-tailed noise, and severely biased noise are also considered. We use the notation $\nu \sim N$ and let Σ_N^2 denote the measurement error covariance which we will report as a percentage of Σ_X^2, the variance of the state X.

4.3.1 Lorenz 63

The Lorenz 63 system given by (4.14) was first derived from a Galerkin projection of Rayleigh-Bernard convection and has since become a canonical teaching example for nonlinear dynamical systems [95].

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z. \end{aligned} \tag{4.14}$$

We generated a test dataset by simulating the Lorenz 63 system for 2500 timesteps using timestep length $h = 0.02$, parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, and initial condition $(5, 5, 25)$. Several types of measurement error are added to the numerical solution to test the accuracy of the proposed method. Figure 4.1 shows the data, state, and state estimate for the Lorenz 63 system for measurement noise distributed according to $\nu \sim \mathcal{N}(\mu, \Sigma_X^2)$ with $\mu = (5, -5, -5)$. An initial transient exhibits considerable error but the remainder of the state estimate tracks the chaotic trajectory precisely.

A summary of results using root mean square error (RMSE) for the Lorenz 63 system with various noise distributions in shown in figure 4.2. In each case, noise is set to have

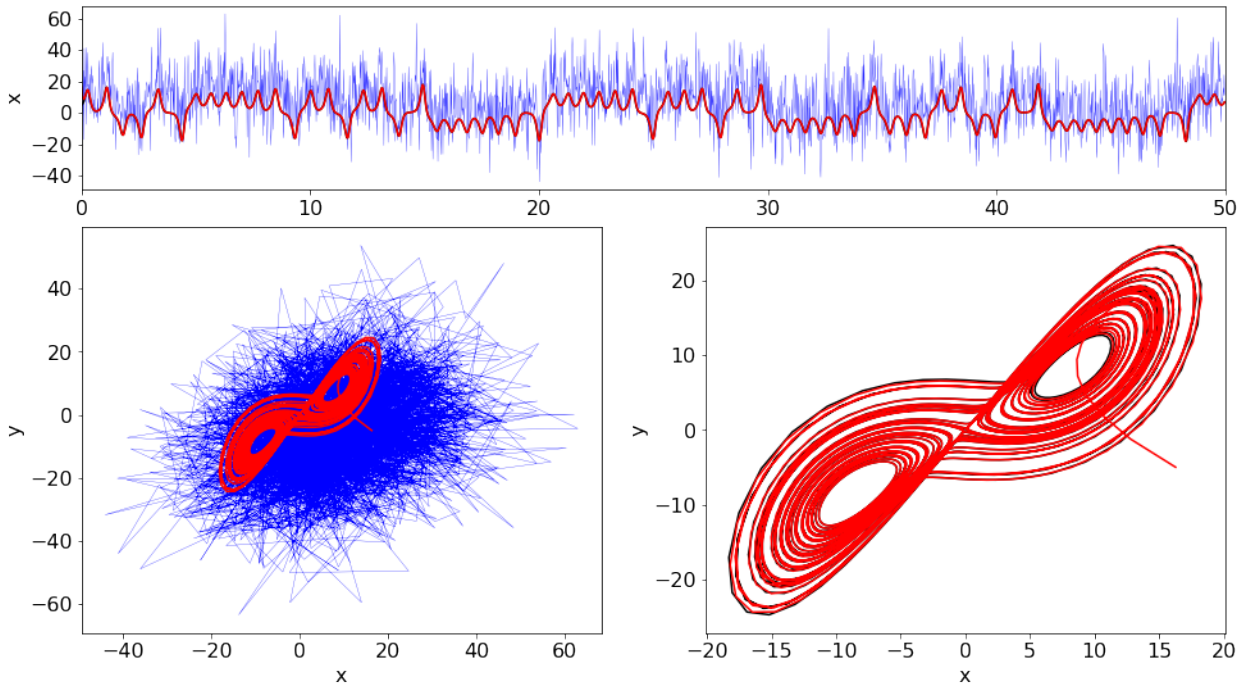


Figure 4.1: Denoising results for Lorenz-63 system corrupted by white noise with non-zero mean. $\nu \sim \mathcal{N}(\mu, \Sigma_X^2)$ with $\mu = (5, -5, -5)$. RMSE = 0.397

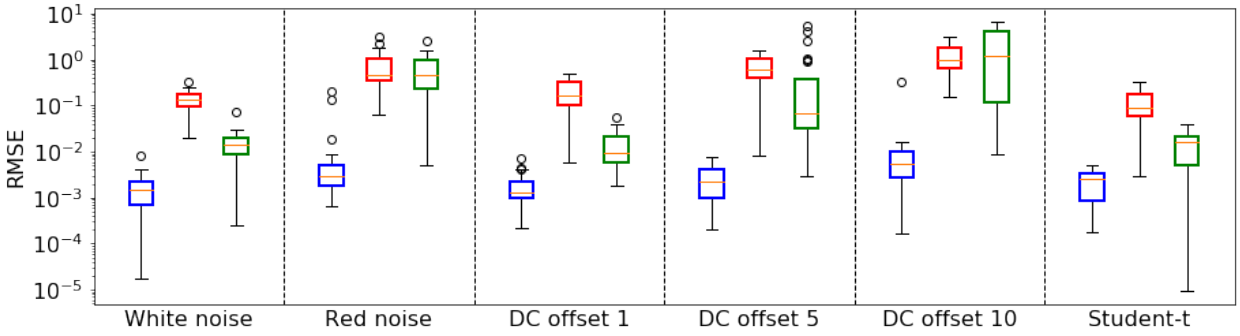


Figure 4.2: Denoising results for Lorenz-63 system with $\sigma_\nu^2 = \sigma_X^2$ according to various distributions. Blue: Runge-Kutta denoising with known dynamics, Red: Runge-Kutta denoising with unknown (μ, σ, ρ), Green: Ensemble RTS Smoother with $N_e = 500$.

variance equal to the data X. We test the method for measurement error distributed according to mean zero Gaussian (white) noise, and Gaussian noise with non-zero mean having magnitude equal to 1, 5, or 10. We also test the case where noise is autocorrelated in time (red noise) according to,

$$\begin{aligned} \nu_{j+1} &= \rho\nu_j + \sqrt{1-\rho^2}\epsilon_j \\ \epsilon_j, \nu_0 &\sim \mathcal{N}(0, \Sigma^2) \end{aligned} \tag{4.15}$$

where $\rho = 0.75$. The last case corresponds to an Ornstein-Uhlenbeck process and results in deviations from the true trajectory that are not remedied by moving average smoothing techniques.

In each case we test the method introduced in this work with and without knowing the parameters ρ, σ, and β. For comparison, we also obtain state estimates using the EnRTS smoother with initial state mean error covariance given by (x_0, Σ_X^2), and process noise given by a mean zero normal distribution with covariance $dt^8 I$ given the use of a fourth order timestepper. We note that the inclusion of some error in from the timestepping scheme did not have significant effects on results. The EnRTS smoother did not perform parameter estimation. In each case the method presented in this work outperformed the

EnRTS smoother when dynamics were known. In the cases where the parameters were unknown, the proposed method resulted in a median RMSE comparable to the EnRTS smoother with known parameters but generally having lower variance.

4.3.2 Lorenz 96

The Lorenz 96 system was introduced by Lorenz as a test model for studying predictability in atmospheric models [96] and has since been shown to exhibit chaotic behavior [78]. it is given by,

$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{4.16}$$

where F is a forcing parameter and $i = 1, \ldots, 40$ with period boundary conditions. The degree of chaotic behavior is determined by F, with $F = 8$ resulting in highly chaotic and $F = 16$ resulting in turbulent behavior [141].

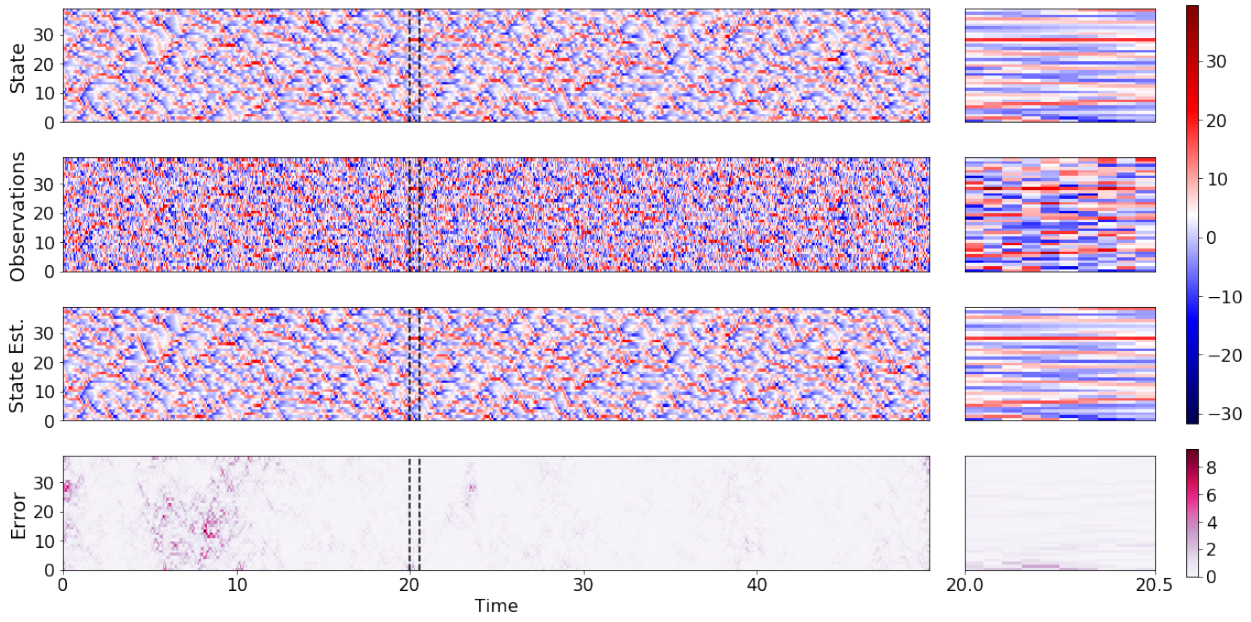


Figure 4.3: Denoising and parameter estimation results for Lorenz-96 system with 100% white noise. $\hat{F} = 15.94$

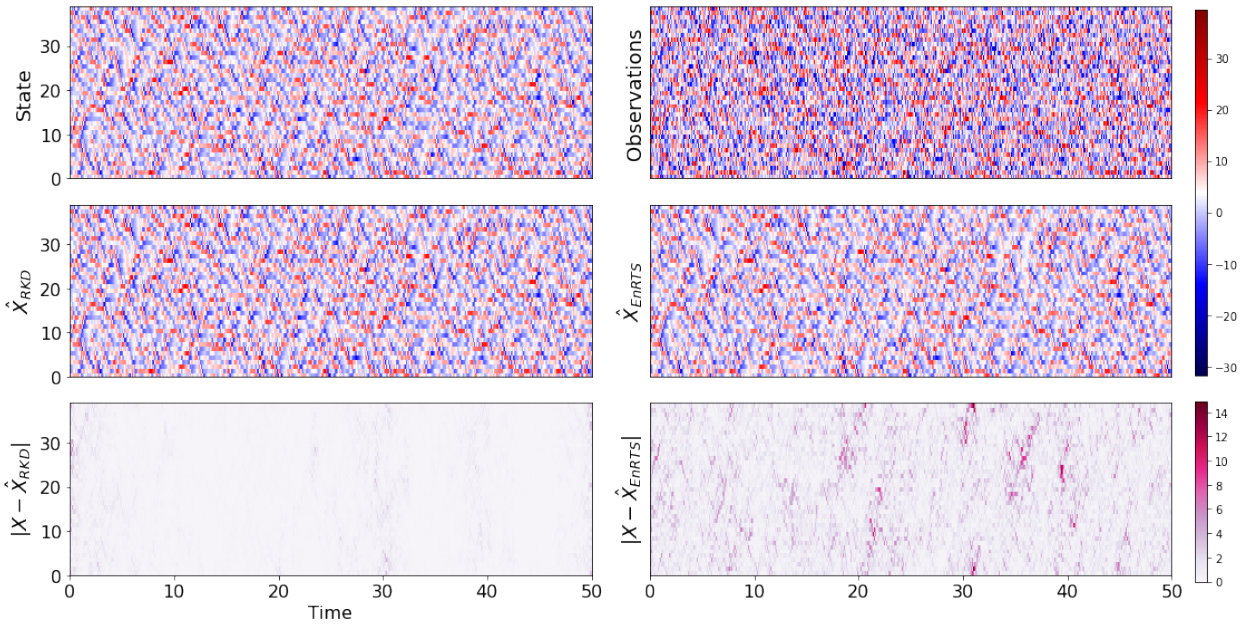


Figure 4.4: Comparison between RKD and EnRTS for Lorenz-96 system with 100% white noise. Top left: X. Top right: Y. Center left: X_{RKD}. Center right: X_{EnRTS}. Bottom left: $|X - X_{RKD}|$. Bottom right: $|X - X_{EnRTS}|$.

Figure 4.3 shows the results of the Runge-Kutta denoising algorithm on the Lorenz 96 system in the turbulent regime where the algorithm is not given the value of F. A more detailed view of state, observations, and state estimate is shown for time $t \in [20, 20.5]$. The learned value of F is 15.94, 0.375% below the true value of 16.

Figures 4.4 and 4.5 compare the results of the proposed algorithm to the EnRTS smoother on a single example of the Lorenz 96 system with known $F = 16$. Absolute error for the Runge-Kutta based smoothing is considerably lower than the EnRTS smoother. The proposed method is also more accurate when it is required to learn the forcing parameter. Figure 4.6 shows the mean and standard deviation of the RMSE for Runge-Kutta based smoothing with and without known F, and the EnRTS smoother applied to the Lorenz 96 system in the turbulent regime across noise levels from $\sigma_N^2 = 0.01\sigma_X^2$ to $\sigma_N^2 = \sigma_X^2$. In each case the Runge-Kutta smoothing outperforms the EnRTS smoother even if F is unknown.

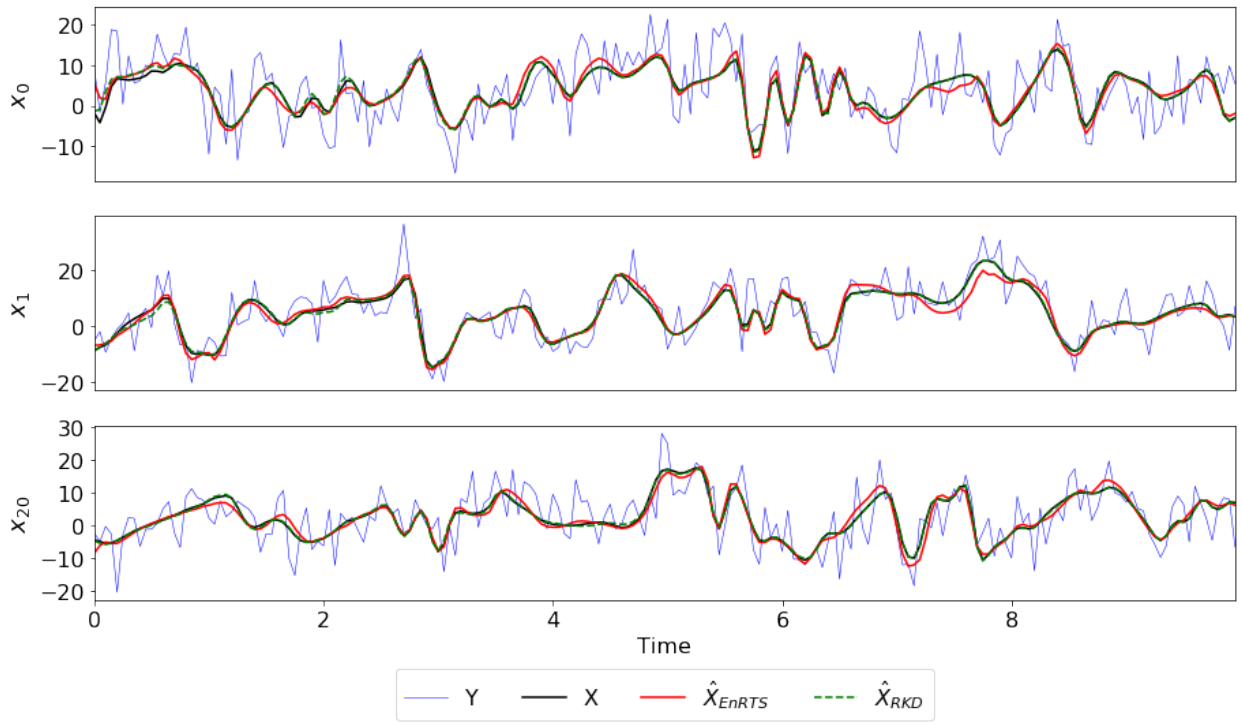


Figure 4.5: Example of data and smoothed time series using RKD and EnRTS for Lorenz-96 system with 100% white noise.

Results for the Lorenz 96 system are only presented with measurement noise coming from a mean-zero Gaussian distribution. If instead red noise was added to the state then both the proposed Runge-Kutta based smoothing and the EnRTS smoother failed to obtain accurate state estimates.

4.3.3 Kuramoto-Sivashinsky

The Kuramoto-Sivashinsky (KS) equation is a fourth order partial differential equation given by (4.17). It is considered a canonical example of spatio-temporal chaos in a one-dimensional PDE [68, 2] and is therefore commonly used as a test problem for data-driven algorithms. The KS equation is a particularly challenging case for filtering algorithms due to it's combination of high dimensionality and nonlinearity [69].

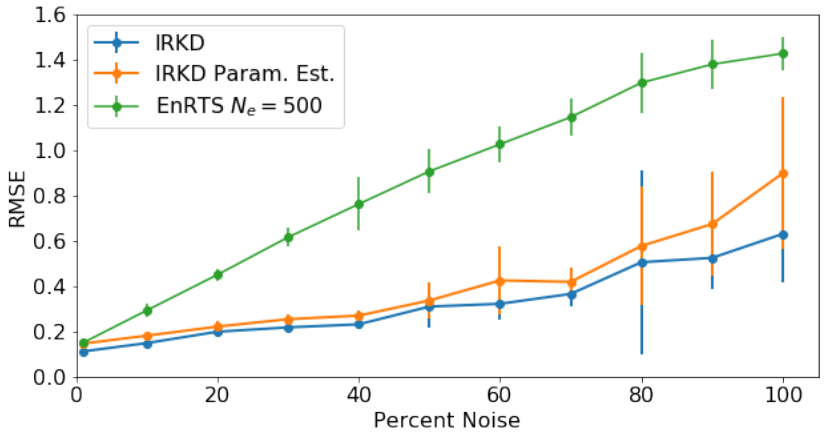


Figure 4.6: Error for denoising Lorenz-96 system with 100% white noise using EnRTS, RKD with known parameters, and RKD with parameter estimation.

$$u_t + uu_x + u_{xx} + u_{xxxx} = 0. \tag{4.17}$$

We solve (4.17) on $[0, 32\pi]$ with periodic boundary conditions using the method outlined in [79] from $t = 0$ to $t = 150$. Figures 4.7 and 4.8 show the results of the proposed method on the Kuramoto-Sivashinsky equation with red noise having $\Sigma_N^2 = 5\Sigma_X^2$ and $\rho = 0.75$.

4.3.4 Optimization over velocity space

In Sec. 4.2.2 we suggest that optimization over incremental values of the state using equation (4.11) is superior to optimizing over incremental velocities as is standard in Runge-Kutta methods. One possible reason for the difference is the potential for large differences in magnitude between $\mathbf{x}$ and $\dot{\mathbf{x}}$. If so, the scale difference could be remedied by scaling time by $\tau = \alpha t$ so that $\|\mathbf{x}\| \approx \|\mathbf{x}_\tau\| = \alpha^{-1}\|\dot{\mathbf{x}}\|$.

Figure 4.9 investigates the performance of the proposed smoothing technique using equation (4.7) (red) and equation (4.11) (blue). Indeed, there exist scalings α such that

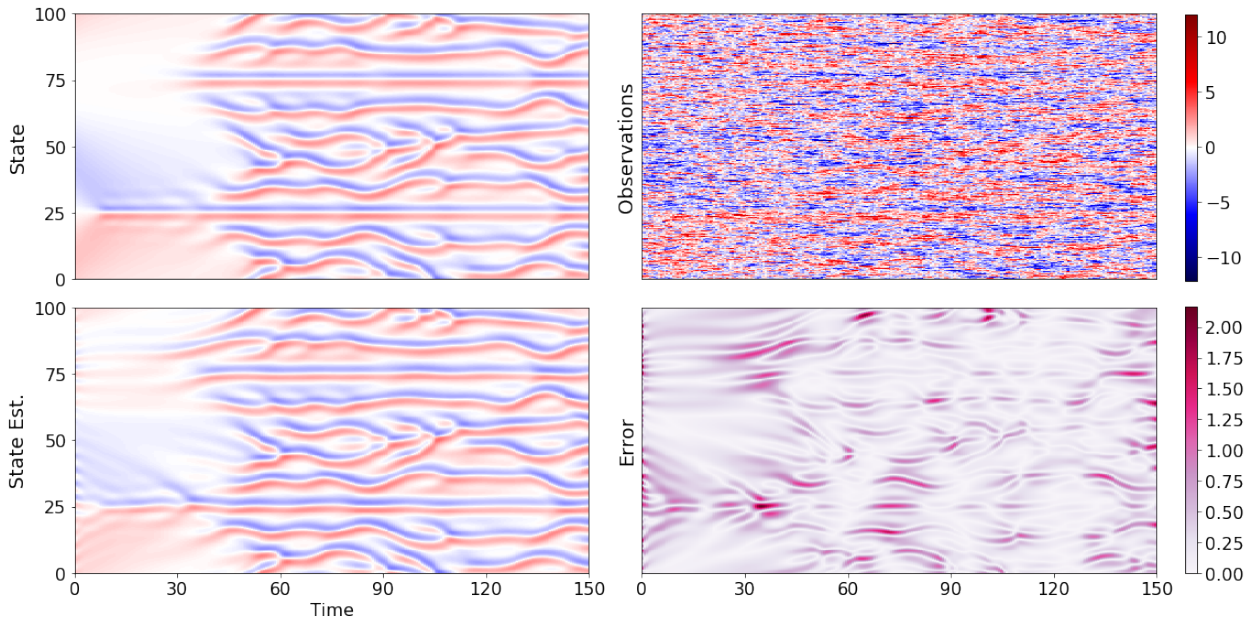


Figure 4.7: Results on Kuramoto-Sivashinsky equation. top left: X. Top right: Y with $\sigma_N^2 = 5\sigma_X^2$. Bottom left: $\hat{X}_{RKD}$. Bottom right: $|X - \hat{X}_{RKD}|$.

optimization over velocity space performs comparatively with optimization over state space. However, finding the optimal scaling may not be possible for a naive application of the smoothing technique where the true state is unknown. Figure 4.9 indicates that for too small of a time dilation α the method's performance exhibits substantial variability across trials.

Sensitivity to parameter for fidelity to measured data

In equation (4.11), measurements are only included in the term $g(\hat{\mathbf{X}}-\mathbf{Y})$. We use either the ℓ^1 norm or squared ℓ^2 norm as a means of pinning the state estimate to the measured data with a constant λ dictating the importance of this difference relative to the terms measuring fidelity to the dynamics. Without the term $g(\hat{\mathbf{X}} - \mathbf{Y})$, the data $\mathbf{Y}$ would only appear in the optimization problem as an initial guess for the solver. A reasonable heuristic for setting λ would be to balance the expected loss due to measurement noise $g(\hat{\mathbf{N}})$ for some

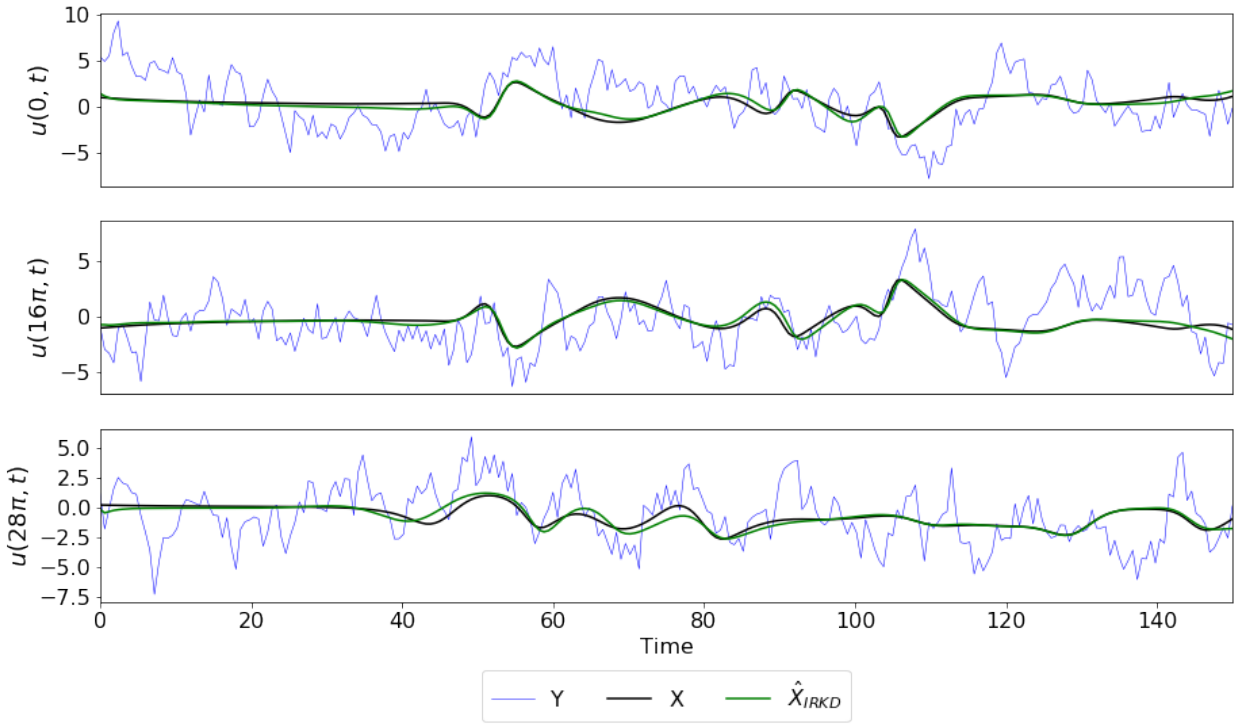


Figure 4.8: Example of data and smoothed time series using RKD for KS system with 500% red noise ($\rho = 0.75$).

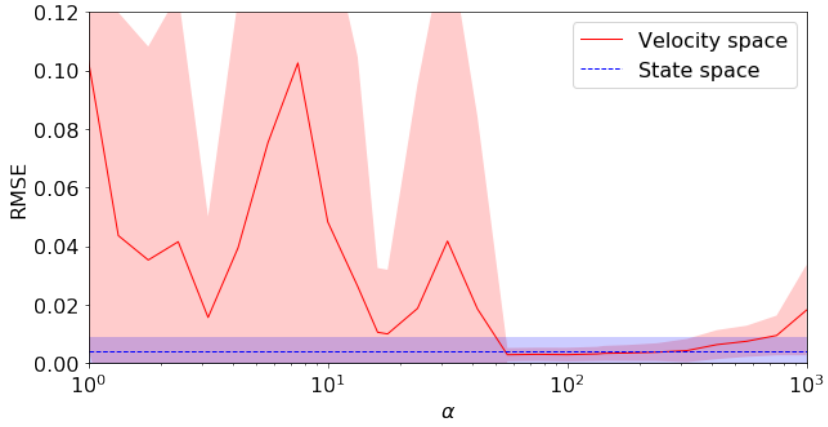


Figure 4.9: Root mean square error for the solution of (4.7) across 25 trials for various values of α in red. RMSE for (4.11) shown in blue.

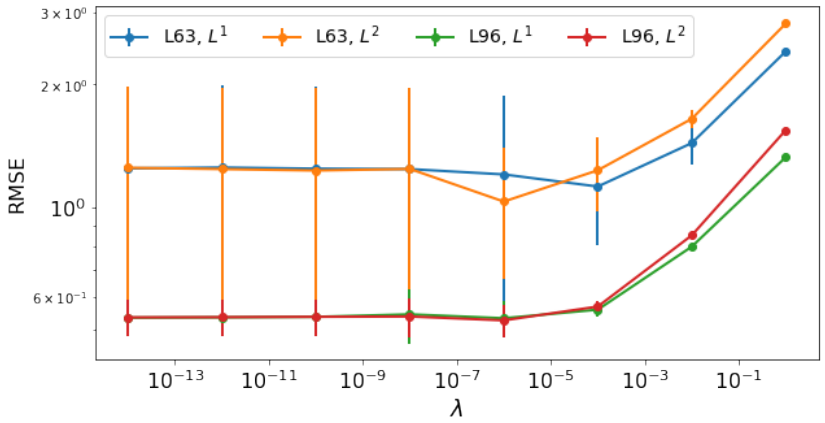


Figure 4.10: Root mean square error for the solution of (4.11) using $g(\cdot) = \lambda\|\cdot\|_1$ or $g(\cdot) = \lambda\|\cdot\|_2^2$ for various λ averaged over 10 random noise samples on the Lorenz 63 and Lorenz 96 systems. λ on the order of $10^{-8} - 10^{-4}$ provides small advantage over $\lambda = 0$ solution.

reasonable guess of the noise $\mathbf{N} = [\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m]$ with the expected loss due to timestepper error, which can be roughly approximated by $m(s+1)dt^p$ for an order-p timestepping scheme having s-steps. However, the method is generally very robust to changes in λ so a careful analysis may not be necessary.

Figure 4.10 shows the sensitivity of the algorithm to changes in the weighting λ of $g(\cdot)$ in the cases where either an ℓ^1 or ℓ^2 norm is used for the Lorenz 63 and Lorenz 96 systems. Surprisingly, the method generally performed well even when $\lambda = 10^{-18}$, essentially negating the role of data as anything other than an initial guess. However, this result may not be general and slight improvements in performance were observed for $\lambda \in [10^{-8}, 10^{-4}]$. Each example shown in the work used $\lambda = 10^{-8}$.

4.4 Discussion

In this chapter we have presented a novel technique for fixed interval smoothing and parameter estimation for high dimensional and highly nonlinear dynamical systems. The proposed method forces adherence to known governing equations by minimizing the residual of a Runge-Kutta scheme while also pinning the state estimate to observed data. While more limited than sequential Bayesian filters and smoothers, the method is robust to substantial measurement error and yields accurate results in several cases where Bayesian filters fail. The proposed methodology is shown to be robust to non-Gaussian, offset, and temporally autocorrelated noise even in the high dimensional setting.

Surprisingly, error for the state estimates of the Lorenz 63 and Lorenz 96 system are both low even when no term in the cost function ties it to data. In this case, measurements only appear as an initialization for the optimization of equation (4.11) where $g(\cdot) = 0$. That state estimates are still accurate seems to imply that the optimization starts in the attracting region of a local minima which is the correct state estimate.

We believe that the optimization framework presented in this work suggests future research directions in the fields of system identification, parameter estimation, and filtering. Directions could include extensions of the current work to include nonlinear or partial observation functions as well as considerations of stochastic dynamics.

Chapter 5

DATA-DRIVEN SURROGATE MODEL OF MUSCLE DYNAMICS

This chapter develops a data-driven surrogate model for the half-sarcomere under periodic length changes and activation. The chapter is organized as follows: In Sec. 5.1 we discuss some background on models of the half-sarcomere and motivation for the data-driven approach. In Sec. 5.2 we formulate the goals of the data-driven model in the context of prior work using mechanistic simulations, introduce our computational framework, and discuss training methods. Section 5.3 shows some results applying the data-driven model to novel parameter regimes. Section 5.4 provides a brief discussion.

5.1 Introduction

Since it's discovery in 1954 by A.F. Huxley *et al.* [66, 65], the sliding filament model for skeletal muscle has motivated a series of mathematical descriptions. The sliding filament model describes small units of muscle tissue called sarcomere that exist in chains and are responsible for generating force. Sarcomeres are composed of overlapping filaments running between two disks. Thin filaments attached to either of the disks are called actin filaments. They overlap with thicker myosin filaments in the center of the sarcomere. Myosin filaments are equipped with crossbridges that act together as ensembles of molecular motors to pull the actin filaments towards the center of the sarcomere, thus shortening its length.

Mathematical models for muscle contraction based on sliding filament theory were proposed in 1957 [64]. Since the sarcomere is symmetric, it suffices to model the half-sarcomere. Early models considered actin and myosin filaments to be inextensible. In this case, force generated by the sarcomere is simply the summed forces generated of

each crossbridge. Assuming a continuum limit the crossbridge's density along the length of the myosin filaments and binding sites along actin filaments, one can derive partial differential equation models for the sarcomere [64].

Following early modeling work assuming rigid filaments, experimental evidence suggested that filaments do in fact exhibit compliance [67, 168]. This discovery has several implications for any mathematical model of the half sarcomere. Net axial force is no longer simply the sum of all forces created by crossbridges but instead must be slightly lower due to tension along the stretched filaments. Furthermore, local deformation of filaments may lead to coupling between crossbridges. Indeed, experimental evidence suggests a highly nonlinear relationship between filament overlap, which corresponds to the number of bound crossbridges, and force [58].

Capturing filament compliance with PDE models is nontrivial, though the original Huxley model has been adapted to include extensible filaments [110]. The adapted PDE based theory, however, does not consider mechanical coupling between crossbridges. Monte Carlo techniques have subsequently been proposed that explicitly track the location of each crossbridge and binding site, as well as crossbridge states [38, 158]. These models started with single filament pair interactions [38] and were extended to consider radial geometry of the sarcomere [158]. In contrast to PDE based models, Monte Carlo simulations have revealed that compliance based mechanical coupling does play a role in crossbridge dynamics.

Macroscopic and tissue scale models of muscle will require large scale coupling of individual sarcomere models. Coupling of small numbers of sarcomeres has been explored in the differential equations model case [22, 154]. Coupled models explicitly including effects from filament compliance require scaling already expensive computational models and are thus infeasible. Recent work has looked at coupled ordinary and partial differential equation models that mimic results of Monte Carlo simulations [169] but has not been extended to the multiple sarcomere case. The prohibitive cost of scaling Monte Carlo based simulations motivates surrogate models that allow for faster computation of the in-

put output dynamics of the Monte Carlo simulations with lower computational expense.

This chapter constructs a data-driven surrogate model for the half-sarcomere meant to replicate the behavior of the spatially explicit Monte Carlo techniques discussed in [158, 175, 176]. Each simulation of the model is parameterized by an externally imposed length. Binding sites on each actin filament have permissiveness modulated by calcium, which binds to troponin molecules on the filaments inducing a conformal change and allowing binding. In contrast to past methods which represent dynamics as differential equations, we use a discrete model to track the probability of crossbridges being in one of three bound states throughout the simulations. Transition probabilities between states are modeled in a data-driven manner from simulation data using the Monte Carlo technique outlined in [158]. We are specifically interested in the case of periodically lengthened and activated half-sarcomeres. The goal of this work is to accurately model forcing time series in response to a variety of input parameter regimes in a way that generalizes to unseen inputs.

5.2 Methods

In this section we more formally discuss the problem of building a surrogate model for force calculations and provide detailed explanations for the computational techniques used. The section is organized as follows: in 5.2.1 we motivate the problem of a reduced cost surrogate model using an abstract notation of the full scale mechanistic model presented in [38] and also discussed in [158, 175, 176]. In the 5.2.3 we provide a detailed description of the computational techniques used in this work. In all that follows we will be discussing a dynamical system evolving in time with discretized increments t_j. The notation $\bullet^{(j)}$ will always refer to the quantity $\bullet$ evaluated at time t_j.

5.2.1 *Mathematical formulation*

We first consider an abstract motivation of the surrogate model. The half-sarcomere consists of a bundle of actin and myosin filaments on which we impose periodic length changes and calcium based modulation of binding site permissiveness. We denote the state of the half-sarcomere at time t, including locations of each cross-bridge, their bound state, and locations of each binding site as $\mathbf{x}(t) \in \mathcal{X}$. Input parameters, including actin permissiveness and externally induced length changes are denoted by $\mathbf{u}(t) \in \mathcal{U}$. A single step of the mechanistic simulation described in [38] is given by,

$$\mathbf{x}^{(j+1)} = \phi\left(\mathbf{x}^{(j)}, \mathbf{u}^{(j)}, \boldsymbol{\omega}^{(j)}\right) \tag{5.1}$$

where $\boldsymbol{\omega}^{(j)} \in \Omega$ is a random variable capturing the simulation's stochasticity. In the Monte Carlo simulation, $\boldsymbol{\omega}$ is a number of identical independently distributed uniform random variables used to compute bound states of each cross-bridge. In this work we restrict our attention to periodic input. Let $\mathcal{P}_\tau = \{\{\mathbf{u}^{(j)}\}_{j=1}^m : \mathbf{u}(t) = \mathbf{u}(t+\tau)\,\forall\, t\}$ be the space of length m discretely sampled time series of τ periodic signals and $\mathbf{u} = \{\mathbf{u}^{(j)}\}_{j=1}^m$. The full length mechanistic simulation in [38] is then a function $\Phi : \mathcal{X} \times \mathcal{P} \times \Omega \to \mathcal{X}^m$ given by,

$$\begin{aligned}\{\mathbf{x}^{(j)}\}_{j=2}^m &= \Phi\left(\mathbf{x}^{(1)}, \mathbf{u}, \boldsymbol{\omega}\right) \\ &= \left\{\phi\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}, \omega_1\right), \phi^2\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1:2)}, \boldsymbol{\omega}^{(1:2)}\right), \ldots, \phi^{m-1}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1:m-1)}, \boldsymbol{\omega}^{(1:m-1)}\right)\right\},\end{aligned} \tag{5.2}$$

where $\boldsymbol{\omega}^{(1:l)} = \{\boldsymbol{\omega}^{(j)}\}_{j=1}^l$ and,

$$\phi^l\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1:l)}, \boldsymbol{\omega}^{(1:l)}\right) = \phi(\phi^{l-1}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1:l-1)}, \boldsymbol{\omega}^{(1:l-1)}\right), \mathbf{u}^{(l)}, \boldsymbol{\omega}^{(l)}) \tag{5.3}$$

for $l > 1$. Equation (5.2) allows one to sample trajectories of $\mathbf{x}$ given a prescribed input parameterization $\mathbf{u}$. In this research we are interested in the work generated by the half sarcomere. Letting $f : \mathcal{X} \to \mathbb{R}$ denote the state-dependent force generated by the half-

sarcomere, work over one loop of the periodic forcing is,

$$\begin{aligned} W\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}\right) &= -\mathbb{E}_{\boldsymbol{\omega}}\left[\int_0^{\tau} f(\mathbf{x}(t)) \cdot \dot{z}(t)\, dt\right] \\ &\approx -\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{j=1}^{m_\tau} f\left(\phi^j\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1:j)}, \omega^{(1:j)}\right)\right) \cdot \left(z^{(j)} - z^{(j-1)}\right)\right], \end{aligned} \tag{5.4}$$

where $z(t)$ is the externally imposed sarcomere length encoded in $\mathbf{u}$, $m_\tau = \tau/dt$, and the negative sign has been added since force is computed in the negative z (contraction) direction. Evaluation for the full model Φ requires sampling from Ω as well as a complex numerical optimization problem for each step to balance forces along the length of each filament. In some cases, for example when evaluating the work generated by a new parameter regime, it is useful to use a lower fidelity model with decreased computational cost. Critically, any such model must also maintain sufficient resolution to accurately compute the force exerted by the half-sarcomere at each timestep in the simulation. In this work, we construct a course graining $\mathcal{Y}$ of the space $\mathcal{X}$, function $\tilde{\phi} : \mathcal{Y} \times \mathcal{U} \to \mathcal{Y}$, and $\tilde{f} : \mathcal{Y} \to \mathbb{R}$ such that,

$$\tilde{\phi}\left(\mathbf{y}(\mathbf{x}), \mathbf{u}\right) \approx \mathbb{E}_\omega\, \mathbf{y}\left(\phi(\mathbf{x}, \mathbf{u}, \omega)\right) \tag{5.5}$$

and

$$\tilde{f}(\mathbf{y}(\mathbf{x}), \mathbf{u}) \approx f(\mathbf{x}) \tag{5.6}$$

both hold with negligible residuals for all $\mathbf{x}, \mathbf{u}$ in a large training set. That is, we want a dynamical model $\tilde{\phi}$ of the course grained space that is consistent with the full model ϕ and a consistent means $\tilde{f}$ for computing force from the course grained space.

The fact that $\mathbf{u}$ is in the domain of $\tilde{f}$ and not f implicitly allows for unresolved state variables to affect the force computed from the course grained space, so long as they react quickly to $\mathbf{u}$. Therefore, the function $\tilde{f}$ may be considered to include the effects of important variables for forcing that are skipped over in the course graining $\mathbf{y}$. A canonical example is crossbridge length, which is modulated by changes in sarcomere length.

5.2.2 *State space of the mechanistic simulation*

We provide a brief description of the state space $\mathcal{X}$ of the mechanistic simulation Φ used as a high fidelity model and refer the reader to [158, 175, 176] for more detail. The simulation exploits the symmetry of a sarcomere to focus modeling on the span between the end of the sarcomere, called the Z-disk, and center, called the M-line. A grid of actin and myosin filaments is arranged according to the multi-filament geometry discussed in [158]. Spatial locations of each crossbridge and binding site are explicitly tracked throughout the simulation. We enumerate the crossbridges by which myosin filament they lie on and their location on that filament, and likewise for binding sites on actin filaments. The state space of the mechanistic simulation is uniquely characterized by

$$\mathbf{x}^{(j)} = \left\{ \left\{ \left\{ c_{i,k}^{(j)} \right\}_{k=1}^{n_c} \right\}_{i=1}^{n_M}, \left\{ \left\{ b_{i,k}^{(j)} \right\}_{k=1}^{n_b} \right\}_{i=1}^{n_A}, \mathbf{G}^{(j)} \right\} \tag{5.7}$$

where n_M and n_A are the number of myosin and actin filaments, n_c is the number of crossbridges per myosin filament, and n_b is the number of binding sites per actin filament. Variables c and b are positions of crossbridges and binding sites, respectively. $\mathbf{G}$ is an adjacency matrix encoding connections between crossbridges and binding sites. It is defined by

$$\mathbf{G}^{(j)} \in \{0, 1, 2\}^{(n_c n_M) \times (n_b n_A)}, \tag{5.8}$$

where $\mathbf{G}_{ik}^{(j)}$ is 1, 2, or 3 if the cross bridge indexed by i is unbound, loosely bound, or tightly bound to the binding site indexed by k at time t_j. This representation of connectivity is a large over parameterization. Since no more than one crossbridge may attach to a given binding site and a crossbridge may only bind to one site, at most one element of any particular row or column of $\mathbf{G}$ may be nonzero. Many entries will necessarily be zero due to the simulation's geometry. However, it will be useful to have a complete characterization for later formula.

All simulations in this work are parameterized by three input time series; the length

of the half sarcomere, lattice spacing, and actin permissiveness. The last variable models binding site affinity modulated by calcium. These are abbreviated as $z^{(j)}$, $l^{(j)}$, and $p^{(j)}$, respectively.

$$\mathbf{u}^{(j)} = \begin{pmatrix} \text{Length } (t_j) \\ \text{Lattice spacing } (t_j) \\ \text{Actin permissiveness } (t_j) \end{pmatrix} = \begin{pmatrix} z^{(j)} \\ l^{(j)} \\ p^{(j)} \end{pmatrix} \tag{5.9}$$

We assume constant volume so that $z^{(j)}{l^{(j)}}^2$ is constant throughout any simulation. Three output variables are recorded throughout the simulation: axial force, radial force, and radial tension. We denote these by

$$f(\mathbf{x}^{(j)}) = \begin{pmatrix} \text{Axial force } (t_j) \\ \text{Radial force } (t_j) \\ \text{Radial tension } (t_j) \end{pmatrix} = \begin{pmatrix} f_a^{(j)} \\ f_r^{(j)} \\ f_t^{(j)} \end{pmatrix}, \tag{5.10}$$

where it is implicitly understood that each is dependent on $\mathbf{x}^{(j)}$. Radial force exists in the plan perpendicular to the sarcomere and is generally very close to zero. We are primarily concerned with accurate computations of axial force throughout the simulation.

The training data is composed of a number of runs of the mechanistic model using many parameter regions. We let $\mathcal{P}_{train} \subset \mathcal{P}$ denote the set of parameter regimes used in the training data. Since we do not vary the initial condition $\mathbf{x}^{(1)}$, each run is characterized by its input parameters $\mathbf{u}$ and random variable $\boldsymbol{\omega} \in \Omega^m$. We denote the collection of random variables used for each parameter regime $\mathbf{u}$ as $\Omega_{\mathbf{u}} \subset \Omega^m$.

5.2.3 Data-driven half-sarcomere model

We first consider the problem of finding a good set of course grained coordinates $\mathcal{Y}$ on which to construct a data-driven surrogate model. In many fields of computational physics such as fluid dynamics, there are established means of finding reduced order bases on which to construct reduced models. The problem here is less straight forward,

since many variables tracked in the mechanistic simulation are categorical and we cannot generally assume that dynamics are constrained to a low dimensional space. We therefore rely on domain knowledge of the half-sarcomere to select a basis.

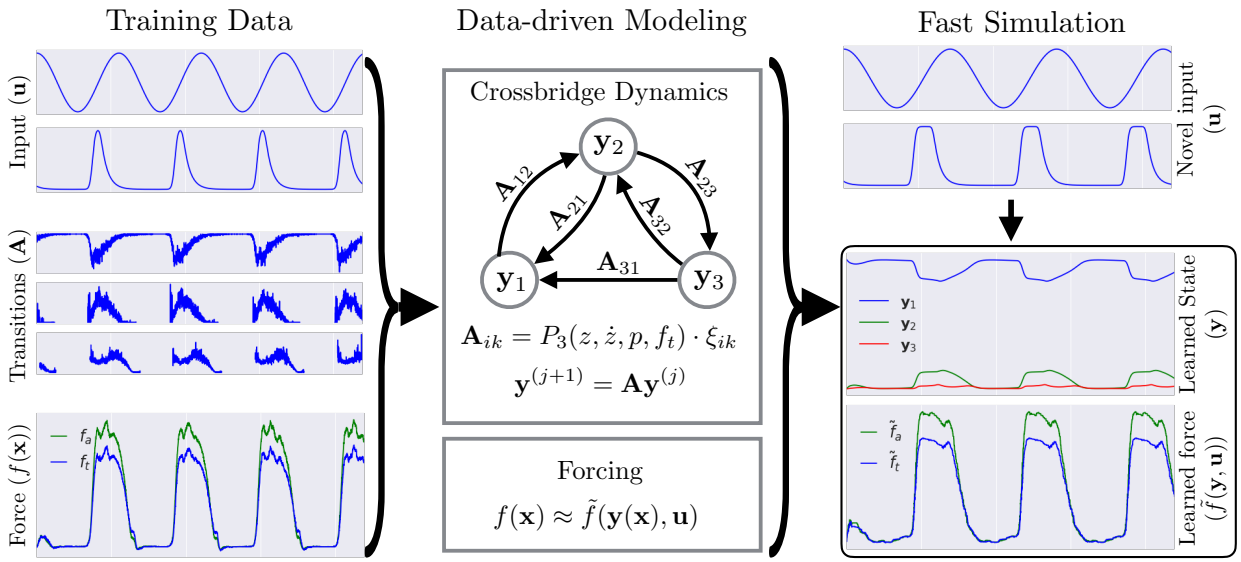


Figure 5.1: Schematic figure for data-driven modeling of muscle. (left) data is collected from MC simulation of full system. (middle) transition probabilities are modeled as a Markov matrix having edge weights dependent on input and radial tension. Forcing is modeled as function of input and number of cross bridges in each state. (right) A new instance of input parameterization is considered, Markov transition probabilities give states of cross bridges, and force and tension are then computed.

The fundamental force generating mechanisms in the actin-myosin interaction are the crossbridges, which in the mechanistic Monte Carlo model exist in one of the states; free, loosely bound, or tightly bound. In our simulations, the mechanistic model accounts for 720 crossbridges. The surrogate model averages over the number of crossbridges in each state to achieve a significant reduction in dimensionality. We will show that simply tracking the fraction of cross-bridges in each state, along with input parameters and forcing, is sufficient to satisfy equations (5.5) and (5.6). The course grained state variables $\mathbf{y}$ and

approximation of force f used in this work are,

$$\mathbf{y}^{(j)} = \begin{pmatrix} \text{Fraction unbound cross-bridges } (t_j) \\ \text{Fraction loosely bound cross-bridges } (t_j) \\ \text{Fraction tightly bound cross-bridges } (t_j) \end{pmatrix} = \begin{pmatrix} y_1^{(j)} \\ y_2^{(j)} \\ y_3^{(j)} \end{pmatrix} \tag{5.11}$$
$$\tilde{f}(\mathbf{y}^{(j)}) = \begin{pmatrix} \text{Approx. axial force } (t_j) \\ \text{Approx. radial tension } (t_j) \end{pmatrix} = \begin{pmatrix} \tilde{f}_a^{(j)} \\ \tilde{f}_t^{(j)} \end{pmatrix}$$

Radial force is omitted from the data-driven model since it is expected to fluctuate in a small region about zero and is not relevant for computing work. Radial tension was found to be important for computing the dynamics of the cross-bridge states.

Dynamics model in course grained space

The internal state of the half-sarcomere is represented only with the fraction of cross-bridges in each of the three bounds states. It is therefore natural to construct a dynamics model based on transition rates between these states. A canonical model for state and input dependent transition rates is given by,

$$\mathbf{y}^{(j+1)} = \tilde{\phi}\left(\mathbf{y}^{(j)}, \mathbf{u}^{(j)}\right) = \mathbf{A}\left(f(\mathbf{y}^{(j)}), \mathbf{u}^{(j)}\right)\mathbf{y}^{(j)}, \tag{5.12}$$

where the constraint $\sum_i \mathbf{A}(\tilde{f}_y(\mathbf{y}^{(j)}), \mathbf{u}^{(j)})_{ij} = 1$ enforces that the total number of cross-bridges in maintained. Since crossbridges do not communicate with each other, it seems intuitive that the dynamics should be linear in $\mathbf{y}$. However, $\mathbf{u}$ alone seems to be insufficient to accurately compute transition probabilities. Indeed evidence suggests that load does affect crossbridge dynamics [166]. We found that including radial tension was important for accurate prediction. Individual transition probabilities are modeled using polynomial regression on features including radial tension and input. Specifically,

$$\mathbf{A}\left(f_y(\mathbf{y}^{(j)}), \mathbf{u}^{(j)}\right)_{ik} = P_3\left(z^{(j)}, \dot{z}^{(j)}, p^{(j)}, f_t^{(j)}\right) \cdot \xi_{ik}, \tag{5.13}$$

where P_q indicates the set of polynomial features of order less than or equal to q in the features and ξ_{ik} is a learned weight vector. The term $\mathbf{A}(f_y(\mathbf{y}^{(j)}), \mathbf{u}^{(j)})_{ik}$ is the probability of a crossbridge transitioning from state i to state k during the timespan $(t_j, t_{j+1}]$.

We impose several constraints on the matrix $\mathbf{A}$. Transitions from unbound (state 1) to tightly bound (state 3) are prohibited in the mechanistic model so $\mathbf{A}_{13}$ is defined to be zero. We also need to ensure each column sums to one and that entries are non-negative. Empirical evidence from the mechanistic simulation suggests that the diagonal entries of $\mathbf{A}$ corresponding to the probability that a crossbridge does not change state are much larger than off-diagonal. We train only off-diagonal entries and subtract those from each column sum to obtain the probability of no transition. To ensure non-negativity we project each transition probability onto $[0, 1]$ after applying (5.13). This strategy could in theory still result in a negative probability if the two learned off-diagonal entries sum to greater than one, but this has not been observed and could easily be flagged during a simulation.

Training the dynamics model

Weight vectors ξ_{ik} are trained using data from the mechanistic model. This requires computing transition probabilities between states at each timestep of each of the simulations used as training data. It is be helpful to consider a simpler notion of crossbridge connectivity than $\mathbf{G}$ where we ignore binding sites and focus only on the bound state of each crossbridge. The time dependent averaged crossbridge state vector $\mathbf{S}^{(j)} \in \{0, 1, 2\}^{n_c n_M}$ is given by

$$\mathbf{S}_i^{(j)} = \max_k \mathbf{G}_{ik}^{(j)}. \tag{5.14}$$

This relates nicely to the course grained variables through the relation

$$y_i^{(j)} = \frac{\left|\left\{k : S_k^{(j)} = i\right\}\right|}{n_c n_M}. \tag{5.15}$$

Empirical transition probabilities for each step of the simulation are given by,

$$\hat{\mathbf{A}}_{ik}^{(j)} = \frac{\left|\left\{\ell : S_\ell^{(j)} = i \text{ and } S_\ell^{(j+1)} = k\right\}\right|}{y_i^{(j)}} \tag{5.16}$$

where $\hat{\mathbf{A}}_{ik}^{(j)}$ is left undefined when $y_i = 0$. The matrix of empirical transition probabilities $\hat{\mathbf{A}}^{(j)}$ will vary across simulations depending on initial condition $\mathbf{x}^{(1)}$, input $\mathbf{u}$, and random variable $\boldsymbol{\omega}$.

Each of the ξ_{ik} is trained using the collection of all timesteps in each simulation across all parameter regimes considered in the training dataset. The numerator of (5.16) is distributed according to $\text{Bin}(y_i^{(j)}, \mathbf{A}_{ik}^{(j)})$ and has mean $\mathbf{A}_{ik}^{(j)}$ and variance ${y_i^{(j)}}^{-1}\mathbf{A}_{ik}^{(j)}(1 - \mathbf{A}_{ik}^{(j)}) \leq (4y_i^{(j)})^{-1}$. We therefore initially tried a regression for ξ_{ik} that weighed each observation $\hat{\mathbf{A}}_{ik}^{(j)}$ by the approximate precision $2\sqrt{y_i^{(j)}}$. However, this method does not yield predictive results. One explanation is that the denominator of (5.16) is highly correlated with the actin permissiveness $p^{(j)}$ and other parameters. Indeed, the mechanistic simulation generally has periodic intervals in which $y_2 = y_3 = 0$.

Instead, we simply threshold observations where $y_i^{(j)}$ is below a fixed tolerance $y_{min} = 5$ and weight the remaining observations equally. Since the cubic polynomial kernel may result in a more ill-conditioned linear system, we impose an ℓ^1 penalty on the weight vector to encourage parsimonious weight vectors, as is common in many physical applications (see Chapter 2 and [18]). To maintain a tractable size for the linear system even with many trials of each parameter region, we average over the empirical transition rates and radial tension within each parameter regime. The resulting cost function is given by,

$$\xi_{ik} = \arg\min_{\xi} \sum_{\mathbf{u},j} \left(\left[\bar{\mathbf{A}}_{ik}^{(j)}\right] - P_3\left(z_j, \dot{z}^{(j)}, p^{(j)}, \bar{f}_t^{(j)}\right) \cdot \xi \right)^2 \delta_{\left(y_i^{(j)} > y_{min}\right)} + \lambda \|\xi\|_1. \tag{5.17}$$

where

$$\begin{aligned}\bar{\mathbf{A}}_{ik}^{(j)} &= \frac{1}{|\Omega_{\mathbf{u}}|}\sum_{\boldsymbol{\omega}\in\Omega_{\mathbf{u}}} \hat{\mathbf{A}}(\mathbf{x}^{(1)},\mathbf{u},\boldsymbol{\omega})_{ik}^{(j)},\\ \bar{f}_t^{(j)} &= \frac{1}{|\Omega_{\mathbf{u}}|}\sum_{\boldsymbol{\omega}\in\Omega_{\mathbf{u}}} f_t(\mathbf{x}^{(1)},\mathbf{u},\boldsymbol{\omega})^{(j)},\end{aligned} \tag{5.18}$$

and δ is a Kronecker delta function indicating when there is sufficient sample size to trust the observation. Each monomial term in the range of P_3 is normalized across the training dataset so that regularization is independent of measurement units. Optimization for (5.17) is performed in the Scikit Learn python library [122]. The parameter λ is found using 10-fold cross validation.

Forcing model in course grained space

Force in the mechanistic simulation is computed by summing over forces generated by individual crossbridges. However, there are properties of the crossbridges not encoded in the averaged bound states. Each crossbridge acts as a spring which is stretched according to the length between it's base on a myosin filament and binding site. This distance changes according to input parameters. Therefore, the data-driven model for forcing is constructed from data as a function of the averaged cross-bridge states as well as the input parameters $\mathbf{u}$.

We construct $\tilde{f}$ as an ensemble of regression trees using gradient boosting [47] implemented in Scikit Learn [122]. The gradient boosting algorithm sequentially constructs an ensemble of regression trees [14] by minimizing the residual error from previous trees and then performing a line search to find an optimal multiplier of the new tree. The resulting $\tilde{f}$ is of the form,

$$\tilde{f}(\mathbf{y},\mathbf{u}) = \sum_{m=1}^{M} h_m(z,(z-z_0)^+,\dot{z},l,y_2,y_3) \tag{5.19}$$

where each $h_m(\mathbf{y},\mathbf{u})$ is a regression tree and $M = 100$. The term $(z-z_0)^+$ is the positive component of the length minus it's temporal mean and allows for distinct functional forms between when length is greater or less than its average. Actin permissiveness and

the number of free crossbridges have been omitted since they do not directly affect forcing. We use the squared error loss function so that each tree h_i is simply trained on the residual error of the sum over all previous trees without line search. Forcing and state fractions used in training $\tilde{f}$ are averaged over runs having the same input parameters. The optimization for each tree is given by,

$$h_i(\mathbf{y}, \mathbf{u}) = \arg\min_{h} \sum_{\mathbf{u},j} \left(\bar{f}\left(\mathbf{x}^{(j)}\right) - \sum_{m=1}^{i-1} h_m\left(\bar{\mathbf{y}}^{(j)}, \mathbf{u}^{(j)}\right) - h\left(\bar{\mathbf{y}}^{(j)}, \mathbf{u}^{(j)}\right) \right)^2 \tag{5.20}$$

where

$$\begin{aligned} \bar{f}^{(j)} &= \frac{1}{|\Omega_{\mathbf{u}}|} \sum_{\boldsymbol{\omega} \in \Omega_{\mathbf{u}}} f(\mathbf{x}^{(1)}, \mathbf{u}, \boldsymbol{\omega})^{(j)}, \\ \bar{\mathbf{y}}^{(j)} &= \frac{1}{|\Omega_{\mathbf{u}}|} \sum_{\boldsymbol{\omega} \in \Omega_{\mathbf{u}}} \mathbf{y}(\mathbf{x}^{(1)}, \mathbf{u}, \boldsymbol{\omega})^{(j)}. \end{aligned} \tag{5.21}$$

Each regression tree h_m is a piecewise constant function that splits the domain along variables that result in minimal variance of the target function in each half of the resulting split domain. This is performed recursively until a maximum depth of 5 is achieved or variance reduction in the splitting is below a threshold.

There are several advantages to using the gradient boosting method. The learned function $\tilde{f}$ is composed of a large ensemble of weak learners which allows the model to be accurate without suffering from overfitting. The resulting $\tilde{f}$ is more accurate than those other machine learning techniques tested for this work, including neural networks. Regression trees also constrain predictions to lie within the same range as the training data so overfitting, if present, will not result in extreme values. The trade of is that regression trees, and even more so ensembles, lack the interpretability of linear or polynomial models. There is considerable room for exploring more interpretable functional forms for $\tilde{f}$ that may provide insight into the mechanics of the model, but doing so is beyond the scope of the current work.

Simulating new data

The data-driven model allows us to compute forcing given new input parameters $\mathbf{u} \in \mathcal{P}_{test}$. The data-driven simulation is given by,

$$\begin{aligned}\{\mathbf{y}^{(j)}\}_{j=2}^{m} &= \tilde{\Phi}\left(\mathbf{x}^{(1)}, \mathbf{u}\right) \\ &= \left\{\mathbf{y}^{(1)}, \tilde{\phi}\left(\mathbf{y}^{(1)}, \mathbf{u}^{(1)}\right), \tilde{\phi}^2\left(\mathbf{y}^{(1)}, \mathbf{u}^{(1:2)}\right), \ldots, \tilde{\phi}^{m-1}\left(\mathbf{y}^{(1)}, \mathbf{u}^{(1:m-1)}\right)\right\},\end{aligned} \tag{5.22}$$

where $\mathbf{y}^{(1)} = \mathbf{y}\left(\mathbf{x}^{(1)}\right)$. The work done by the half-sarcomere over one period of oscillation is computed using the data-driven approximation of force as,

$$\tilde{W}(\mathbf{x}^{(1)}, \mathbf{u}) = \frac{1}{\beta} \sum_{j=\alpha}^{\alpha+\beta m_\tau} \tilde{f}\left(\tilde{\phi}^j\left(\mathbf{y}\left(\mathbf{x}^{(1)}\right), \mathbf{u}^{(1:j)}\right)\right) \cdot \left(z^{(j)} - z^{(j-1)}\right) \tag{5.23}$$

where $\beta = 3$ is an integer number of periods to average over and $\alpha = m - \beta m_\tau$ is a burn in time to allow for decay of any initial transient.

5.3 Results

The dataset used to train the data-driven model consists of 20 sample trajectories from each of 150 unique parameter regimes in $\mathcal{U}$. It is split into 135 training parameter regimes and 15 for model evaluation. Input parameters are altered by changing the period of length changes $[45.45, 55.55, 71.43$ ms$]$, phase offset of calcium activation as fraction of length cycle period $[0, 0.1, 0.2, \ldots, 0.9]$, and half life of calcium activation $[12, 15, 18, 21, 25$ ms$]$. In this section we compare results of the Monte Carlo simulation to those from the data-driven simulation on the parameter regimes reserved for testing. We highlight that quantities all results from the data-driven model are computed only from the initial condition of the mechanistic simulation. Therefore, accuracy is shown not only for point wise computations of transition probabilities, but of the full length time series.

Figure 5.2 shows the results for a single parameter regime of the eight learned transi-

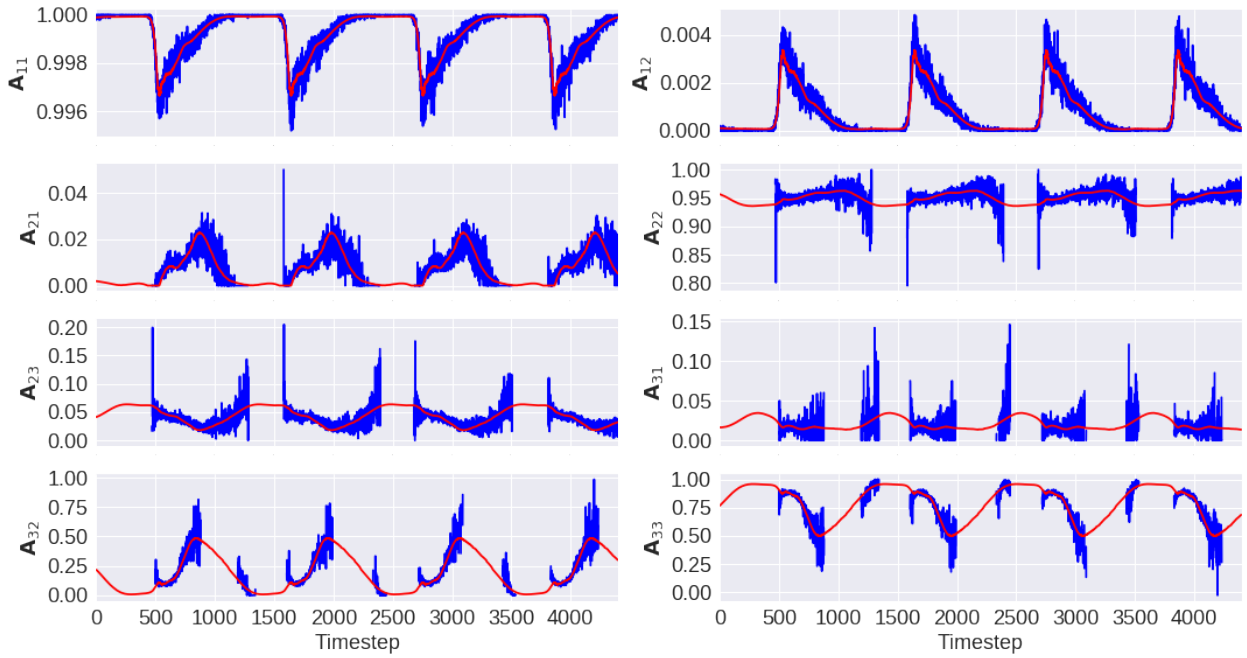


Figure 5.2: Example of empirical transition probabilities from mechanistic model in blue and computed transition probabilities from data-driven model in red for input regime in testing data.

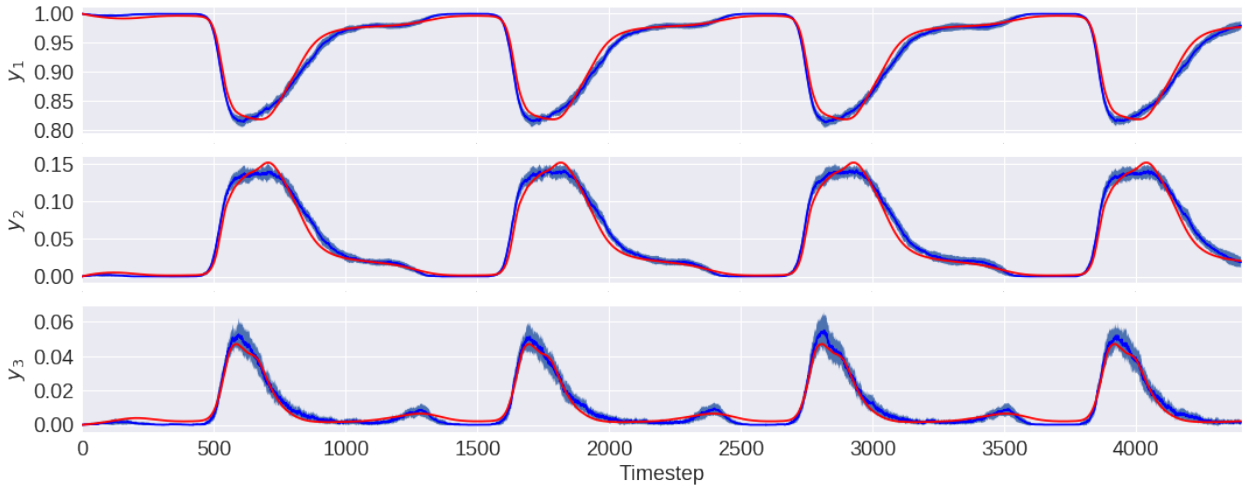


Figure 5.3: Example of crossbridge state averages for the mechanistic model in blue and data-driven model in red for parameter regime in testing data. Mechanistic data is averaged over 20 runs with shaded region indicating one standard deviation.

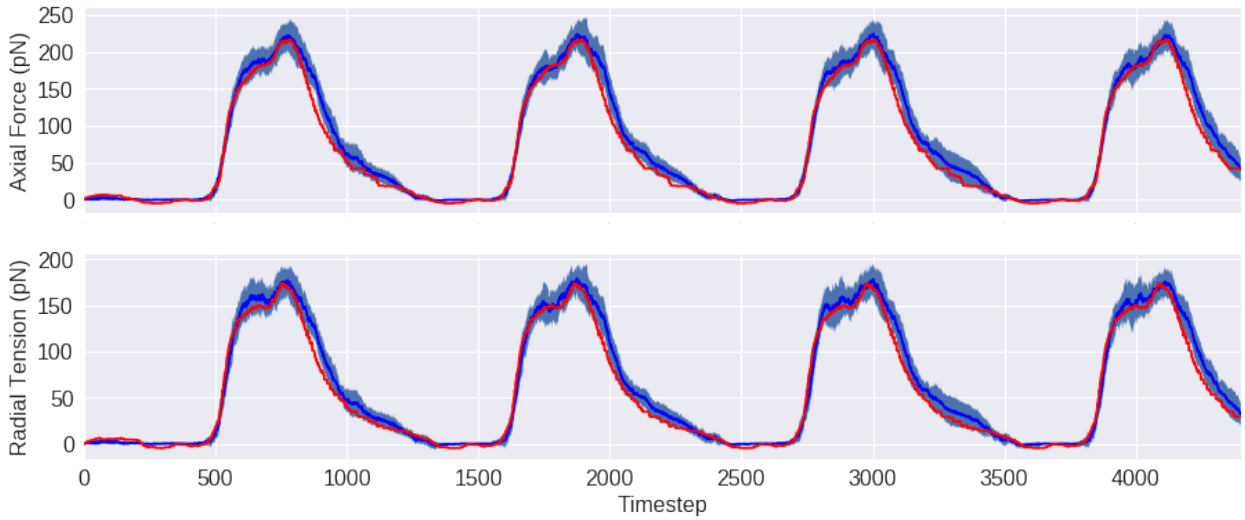


Figure 5.4: Example of forcing for the mechanistic model in blue and data-driven model in red for parameter regime in testing data. Mechanistic data is averaged over 20 runs with shaded region indicating one standard deviation.

tion probabilities, $\hat{\mathbf{A}}_{ik}^{(j)}$, and empirical transition probabilities averaged over all 20 trials, $\bar{\mathbf{A}}_{ik}^{(j)}$. We highlight that empirical transition probabilities are only shown in the plot where they are defined for each of the sample trajectories. Substantially increased variance in certain regions is likely due to low populations in the pre-transition state during those periods.

The crossbridge state densities resulting from the transition probabilities in Fig. 5.2 are shown in Fig. 5.3. Blue curves and regions indicate the mean and one standard deviation band of observed y across all trials of the Monte Carlo simulation for this parameter regime. While slight deviation is observed in the data-driven model, it largely mimics the behavior of the Monte Carlo simulation.

Finally, for the same parameter regime shown in figures 5.2 and 5.3, Fig. 5.4 shows the axial force and radial tension computed using the data-driven model, $\hat{f}^{(j)}$, in red, and from the mechanistic simulation, $\bar{f}^{(j)}$, in blue. Shaded regions indicate one standard deviation bands.

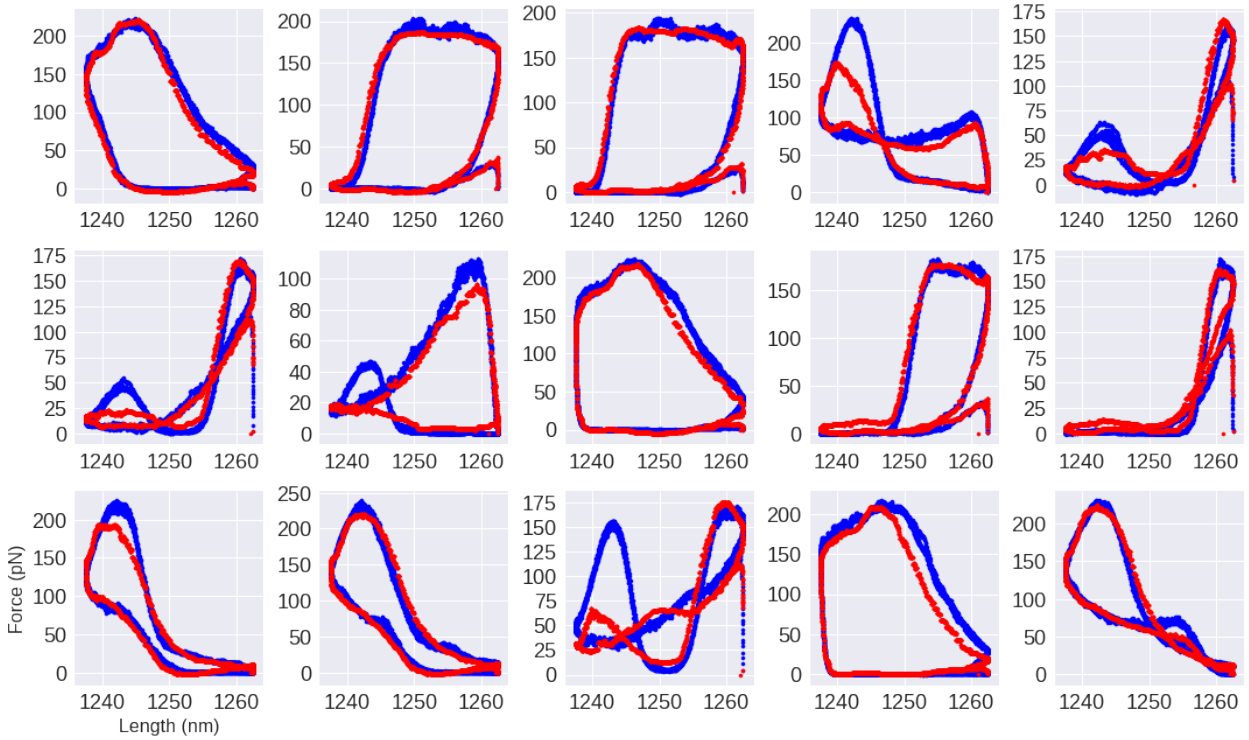


Figure 5.5: Computed workloops for testing data according to mechanistic model in blue and data-driven model in red. Each plot corresponds to an input regime not seen during the training procedure.

Work loops for each of the 15 testing regimes are shown in Fig. 5.5. Red trajectories are generated using the data-driven methods and blue trajectories are given by the average force from the Monte Carlo simulation, $\bar{f}$. A cursory inspection reveals that the data-driven model fails to capture high forcing in the case where there are multiple loops and significant forcing occurs at minimal length. These cases seem to be correlated with high calcium activation during the initial stage of sarcomere shortening.

Figure 5.6 summarizes the key results in this chapter regarding work loop computations. We compare work from the mechanistic (Monte Carlo) model with that from the data-driven model. Parameter regimes included in the training data are shown in blue, and those from the testing data in red. The work computed by the data-driven model does not always fall within the one standard deviation band, but the general trends are maintained.

5.4 Discussion

In this chapter we have developed a novel data-driven technique for modeling the half-sarcomere under periodic length changes and activation. The data-driven model is trained on data from and mimics the behavior of the Monte Carlo techniques developed in [38, 158] at a significantly decreased computational expense. In particular, force traces and work loops are largely consistent when the two models are compared.

While the current work does generally replicate dynamics on the course grained variables and force traces given new input parameters, several regimes in the testing set indicate that some features are missed. In particular, certain increases in axial force during the contracting phase of the sarcomere are underestimated, even when crossbridge state densities are accurate. This indicates that the data-driven model for force may be insufficient. Possible future work could include expanding $\mathbf{y}$ in the hopes of better resolving f or looking in to different anstazes for $\tilde{f}$.

The current work seeks to build towards a scalable model that may be used to run simulations of many sarcomeres coupled end to end. However, in its present state, it

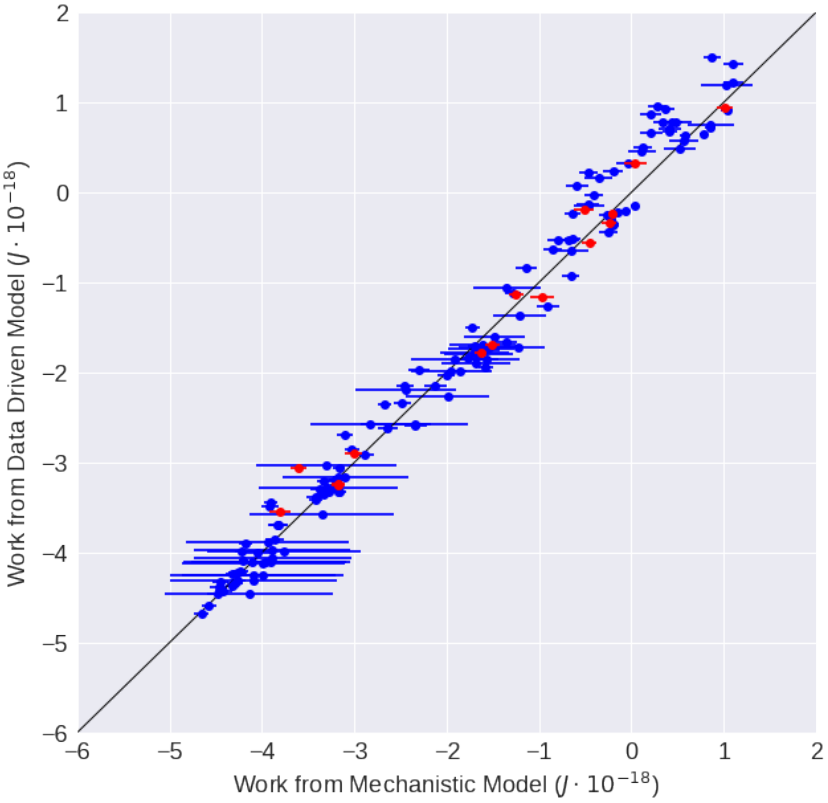


Figure 5.6: Computed work in mechanistic model on x-axis and data-driven model on y-axis. Blue data points correspond to input in training data while red is testing. Error bars indicate one standard deviation around mean value observed over 20 trials of the mechanistic model.

should be considered a proof of concept rather than a finished product. In particular, while we have tested the model on novel parameter regimes, all input regimes in both the testing and training data are periodic. This may not be the case in a more physiological scenario, so the model should be tested on non-periodic data before scaling. Furthermore, the current model computes force generated by the half sarcomere. Scaling would require considering this force as acting on a set of sarcomeres with masses moving in a viscous medium. Tuning for drag coefficients may be a highly nontrivial problem.

In addition to scaling towards multi-sarcomere models, the current work could be

improved on by more careful design of the data-driven functions. We have constructed the framework in this chapter to demonstrate that it is feasible to replicate the behavior of the Monte Carlo simulation with a fast deterministic system, but do not claim that the method presented is optimal. In particular, we suspect that there may be some benefit to modeling the dynamics in the course grained space using non-Markovian dynamics. This is a common technique in reduced order modeling where unresolved variables become memory terms in the dynamics [167, 117].

Chapter 6
CONCLUSIONS

This thesis outlines four contributions to the growing field of data-driven study of dynamical systems. First, we presented a method for identifying partial differential equations from data. The technique builds upon the popular sparse regression framework for nonlinear system identification and has itself been improved upon for many interesting applications. We next developed a framework for neural learning of dynamical systems that is robust to noise. We demonstrate this method on a variety of canonical dynamical systems models and also discuss some challenges associated with interpolating governing equations with neural networks. We also address a method for smoothing high dimensional nonlinear dynamical systems based on adherence to time-stepping schemes rather than using sequential Monte Carlo methods and show that it is effective even in the very high dimensional and nonlinear setting. Lastly, we develop a novel data-driven approach to constructing a fast surrogate model of the half-sarcomere. This conclusion chapter will briefly review each of the four contributions and also suggest a number of future projects in the field.

Sparse identification of partial differential equations

Chapter 2 developed a method for identifying governing partial differential equations from data. The method builds on previous sparse regression techniques for ordinary differential equations, including the Sparse Identification of Nonlinear Dynamics (SINDy) method. Governing equations are assumed to lie in the span of an over-determined basis basis and a library of candidate functions is constructed from measurements. Spatial and temporal derivatives are determined via numerically differentiating the data. Coef-

ficients in the partial differential equation are found using a sparse regression algorithm called sequential threshold ridge regression. The method is demonstrated on a variety of partial differential equations of varying complexity. As a test, we also show that the method accurately recovers the heat equation when discretized probability density functions are constructed from a single realization of Brownian motion.

We also extend the sparse regression framework to allow for identification of partial differential equations with spatially or temporally varying coefficients. By grouping library functions based on timestep or spatial location, we are able to use group sparsity to identify parsimonious representations of the spatio-temporal dynamics while allowing to variability in coefficients. Both the group LASSO and a group hard-thresholding algorithm are implemented on a variety of test problems.

Robust neural network Based Forecasting

Chapter 3 developed a robust method for training neural networks as forecasting models for dynamical systems in the case of noisy and limited data. We explicitly considered noise in a map between observations, rather than assuming idealized trajectories and were able to train both noise estimates and the neural network interpolation of the velocity field simultaneously. The resulting optimization problem was motivated by observing the joint posterior distribution of the measurement noise and neural network parameters, then adjusted to favor accurate predictions over multiple timesteps. This model circumvents the need for alternating between state estimates and training a dynamics model. We applied this technique to several canonical nonlinear dynamics systems at varying levels of noise to demonstrate its effectiveness. The chapter also discusses limitations of neural networks as interpolation methods for dynamical systems. Several examples are shown that indicate over fitting of the velocity field, even if the learned dynamics still accurately forecast certain trajectories. We propose that neural networks may be useful as forecasting in practical setting dynamics are constrained to lie on a bounded attractor since training

data is more likely to densely sample the complete domain of the dynamics.

Smoothing by soft adherence to governing equations

Chapter 4 introduced a novel method for smoothing noisy data with known or partially known governing equations. Rather than using sequential Monte Carlo techniques such as the ensemble Kalman or particle filters, which can exhibit poor performance in the high dimensional and nonlinear setting, we measure fidelity of state estimates to governing equations by the residual on an implicit Runge-Kutta timestepping scheme. The resulting method is shown to perform well on smoothing problems given very high levels of noise on high dimensional and nonlinear problems such as the Lorenz 96 model and Kuramoto-Sivashinsky equation. We stress that the smoothing method presented in this work is less general than sequential Monte Carlo methods given that it applies to deterministic systems. Furthermore, it is set up as an optimization problem where the number of parameters grows linearly with length of time series, rather than solving a number of small problems that grows linearly in time series length, as is the case with Kalman and particle filters. However, the exceptional robustness to noise magnitude, shift, and autocorrelation, as well as the seamless ability to smooth high dimensional nonlinear problems, make the proposed method a meaningful contribution.

Data-driven surrogate model of muscle dynamics

In chapter 5 we develop a data-driven surrogate model for the dynamics of the half-sarcomere. This model achieves the same behavior with respect to force traces as more sophisticated Monte Carlo models at a substantially lower computational cost. The model is built by finding a course grained description of the full state space of the Monte Carlo simulation and learning dynamical models on the course grained space. Data-driven representations of the dynamics in the course grained space are trained using data from the full model. Data-driven models for forcing are also learned, and the result fed back into

the dynamics. In doing so, the model seeks to replicate the effects of filament compliance on macro scale dynamics without explicitly tracking micro scale features. We withhold some input parameter regimes and demonstrate accurate reconstruction of course grained state and force traces using the data-driven model and given only knowledge of the initial condition and input. This work allows for faster computation of the forcing behavior of the half-sarcomere, as well as consistent representations of the course grained state variables. It is therefore promising as a step towards multi-sarcomere or even tissue scale models of skeletal muscle.

6.1 Future Directions

- **Robust learning of partial differential equations.** The methods presented in chapter 2 rely on numerical evaluation of high order derivatives from data and are therefore sensitive to noise. Researching means for making the algorithm more robust are a critical step towards practical implementation. Given the success of the robust method for training neural networks presented in chapter 3, we believe it would be interesting to construct sparse regression methods that simultaneously learn dynamics while removing noise from a dataset.

- **Space-time parametric equations.** In chapter 2 we split data into distinct timesteps or spatial locations in order to find PDE models for each subset of the data, resulting in coefficients that can vary in space or time. However, with a sufficiently fine grid, it seems feasible that one could bin the data by areas localized in space and time to determine a coefficient varying in both space and time with some loss of resolution. Future work could consider spatial bins $\{S_i = [x^{(i)}_{min}, x^{(i)}_{max}]\}_{i=1}^{n_x}$ and temporal bins $\{T_i = [t^{(i)}_{min}, t^{(i)}_{max}]\}_{i=1}^{n_t}$. The block diagonal system of equations (2.21) would then have rows given by $u_t^{(i,j)} = \mathbf{\Theta}(u^{(i,j)})\xi^{(i,j)}$ where $u^{(i,j)} = u|_{S_i \times T_j}$ and have solution $\xi^{(i,j)}$ representing the course grained coefficients for the PDE. This same result may be achievable in a more stable manner by introducing a sparsity term to the work in

[93].

- **Reliability and limitations of neural networks.** While neural networks have become a fundamental tool in system identification where governing equations are not easily represented, development of new algorithms has outpaced more careful meta-analysis and their lack of interpretability introduces new challenges. The limitations of neural networks in the context of modeling dynamical systems are not well established. Theoretical work has shown the ability of sufficiently wide single layer networks to accurately approximate any Borel function [37, 62], but these works are constructive and do not consider the training procedure. It is important to address the limitations of neural networks with regards to their applications in system identification given practical restrictions on data and implementation.

- **Non-sequential smoothing of stochastic dynamics.** The method for smoothing presented in chapter 4 assumes smooth trajectories so that it may use high order timestepping schemes. Similar techniques use the Euler-Maruyama scheme and estimates of process and measurement noise covariance to smooth stochastic time series, though they are constrained by the inability of the Euler scheme to accurately model stiff systems. Preliminary work suggests that splitting time increments into many Euler-Maruyama steps yields accurate and robust smoothing techniques even for high-dimensional chaotic dynamics. Future work could look into smoothing schemes based on stochastic Runge-Kutta schemes and compare to ensemble Kalman and particle filters for stochastic partial differential equations.

- **Fast multi-sarcomere models.** The eventual goal of the work discussed in chapter 5 is to develop models capable of simulating the dynamics of large ensembles of sarcomeres. The current work is capable of producing force traces given prescribed length changes and activation. Future work could use that force output, together with physiologically reasonable values for the mass of each sarcomere and viscos-

ity induced drag to construct a coupled spring-mass-damper system for a multi-sarcomere model. This would massively extend the scope of modeling efforts for skeletal muscle.

BIBLIOGRAPHY

[1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Alejandro Aceves, Hatsuo Adachihara, Christopher Jones, Juan Carlos Lerman, David W McLaughlin, Jerome V Moloney, and Alan C Newell. Chaos and coherent structures in partial differential equations. *Physica D: Nonlinear Phenomena*, 18(1-3):85–112, 1986.

[3] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[4] Hassan Arbabi and Igor Mezic. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

[5] Travis Askham and J Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.

[6] Thomas Bengtsson, Peter Bickel, Bo Li, et al. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and statistics: Essays in honor of David A. Freedman*, pages 316–334. Institute of Mathematical Statistics, 2008.

[7] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[8] Peter Benner, Mario Ohlberger, Anthony Patera, Gianluigi Rozza, and Karsten Urban. *Model Reduction of Parametrized Systems*. Springer, 2017.

[9] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 23:539–575, 1993.

[10] Tyrus Berry and Timothy Sauer. Adaptive ensemble kalman filtering of non-linear systems. *Tellus A: Dynamic Meteorology and Oceanography*, 65(1):20331, 2013.

[11] Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[12] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *PNAS*, 104(24):9943–9948, 2007.

[13] Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of chemical physics*, 148(24):241723, 2018.

[14] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[15] O. Bruno and D. Hoch. Numerical differentiation of approximated functions with limited order-of-accuracy deterioration. *SIAM Journal on Numerical Analysis*, 50(3):1581–1603, 2012.

[16] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(19):1–9, 2017.

[17] S. L. Brunton and B. R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67:050801–1–050801–48, 2015.

[18] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15):3932–3927, 2016.

[19] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz. Compressed sensing and dynamic mode decomposition. *Journal of Computational Dynamics*, 2(2):165–191, 2015.

[20] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism a). *Chaos*, 22(4):047510, 2012.

[21] Russel E Caflisch, Stanley J Osher, Hayden Schaeffer, and Giang Tran. Pdes with compressed solutions. *arXiv preprint arXiv:1311.5850*, 2013.

[22] Kenneth S Campbell. Interactions between connected half-sarcomeres produce emergent mechanical behavior in a mathematical model of muscle. *PLoS computational biology*, 5(11):e1000560, 2009.

[23] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[24] Sheng Chen, SA Billings, and PM Grant. Non-linear system identification using neural networks. *International journal of control*, 51(6):1191–1214, 1990.

[25] Sheng Chen and Steve A Billings. Representations of non-linear systems: the narmax model. *International Journal of Control*, 49(3):1013–1032, 1989.

[26] Shizhe Chen, Ali Shojaie, and Daniela M Witten. Network reconstruction from high-dimensional ordinary differential equations. *Journal of the American Statistical Association*, pages 1–11, 2017.

[27] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[28] Alexandre J Chorin and Paul Krause. Dimensional reduction for a bayesian filter. *Proceedings of the National Academy of Sciences*, 101(42):15013–15017, 2004.

[29] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[30] Charles K Chui, Guanrong Chen, et al. *Kalman filtering*. Springer, 2017.

[31] J. D. Cole. On a quasi-linear parabolic equation occurring in aerodynamics. *Quart. Appl. Math.*, 9:225–236, 1951.

[32] T. Colonius and K. Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197:2131–2146, 2008.

[33] T. Colonius and K. Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197:2131–2146, 2008.

[34] PHILIPPE Courtier, J-N Thépaut, and Anthony Hollingsworth. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387, 1994.

[35] M. Cross and P. Hohenberg. Pattern formation out of equilibrium. *Reviews of Modern Physics*, 65:851–1112, 1993.

[36] James Crutchfield and Bruce McNamara. Equations of motion from a data series. *Comp. sys.*, 1:417–452, 1987.

[37] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[38] Thomas L Daniel, Alan C Trimble, and P Bryant Chase. Compliant realignment of binding sites in muscle: transient behavior and mechanical tuning. *Biophysical Journal*, 74(4):1611–1621, 1998.

[39] Bryan Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nat. comm.*, 6, 2015.

[40] Bryan Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015.

[41] Suddhasattwa Das and Dimitrios Giannakis. Delay-coordinate maps and the spectra of koopman operators. *arXiv preprint arXiv:1706.08544*, 2017.

[42] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[43] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[44] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[45] Geir Evensen and Peter Jan Van Leeuwen. An ensemble kalman smoother for nonlinear dynamics. *Monthly Weather Review*, 128(6):1852–1867, 2000.

[46] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[47] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[48] Crispin Gardiner. *Stochastic methods*. Springer Berlin, 2009.

[49] Matan Gavish and David L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.

[50] Zoubin Ghahramani and Sam T Roweis. Learning nonlinear dynamical systems using an em algorithm. In *Advances in neural information processing systems*, pages 431–437, 1999.

[51] Dimitrios Giannakis. Data-driven spectral decomposition and forecasting of ergodic dynamical systems. *Applied and Computational Harmonic Analysis*, 2017.

[52] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[53] R Gonzalez-Garcia, R Rico-Martinez, and IG Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering*, 22:S965–S968, 1998.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning.

[55] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[57] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*. Springer, 2016.

[58] Hideo Higuchi, Toshio Yanagida, and Yale E Goldman. Compliance of thin filaments in skinned fibers of rabbit skeletal muscle. *Biophysical journal*, 69(3):1000–1010, 1995.

[59] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. In *3rd Annual Allerton Conference*, pages 449–459, 1965.

[60] P. Holmes, J. Lumley, G. Berkooz, and C. Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge, Cambridge, England, 2nd edition, 2012.

[61] E. Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. *Comm. Pure App. Math.*, 3:201–230, 1950.

[62] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[63] PL Houtekamer and Fuqing Zhang. Review of the ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

[64] Andrew F Huxley. Muscle structure and theories of contraction. *Prog. Biophys. Biophys. Chem*, 7:255–318, 1957.

[65] Andrew F Huxley and R Niedergerke. Structural changes in muscle during contraction: interference microscopy of living muscle fibres. *Nature*, 173(4412):971, 1954.

[66] HE Huxley. Changes in the cross-striations of muscle during contraction and stretch and their structural interapretation. *Nature*, 173:149–152, 1954.

[67] Hugh E Huxley, Alex Stewart, Hernando Sosa, and Tom Irving. X-ray diffraction measurements of the extensibility of actin and myosin filaments in contracting muscle. *Biophysical journal*, 67(6):2411–2421, 1994.

[68] James M Hyman and Basil Nicolaenko. The kuramoto-sivashinsky equation: a bridge between pde's and dynamical systems. *Physica D: Nonlinear Phenomena*, 18(1-3):113–126, 1986.

[69] M Jardak, IM Navon, and M Zupanski. Comparison of sequential data assimilation methods for the kuramoto–sivashinsky equation. *International journal for numerical methods in fluids*, 62(4):374–402, 2010.

[70] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[71] J. N. Juang. *Applied System Identification*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1994.

[72] J. N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985.

[73] J-N Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5):620–627, 1985.

[74] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman. Identification of observer/Kalman filter Markov parameters: Theory and experiments. Technical Memorandum 104069, NASA, 1991.

[75] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

[76] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[77] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[78] A Karimi and Mark R Paul. Extensive chaos in the lorenz-96 model. *Chaos: An interdisciplinary journal of nonlinear science*, 20(4):043105, 2010.

[79] Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[80] I. Kevrekidis, C. Gear, J. Hyman, P. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Comm. Math. Sci.*, 1(4):715–762, 2003.

[81] Ioannis G Kevrekidis, C William Gear, and Gerhard Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. *AIChE Journal*, 50(7):1346–1355, 2004.

[82] Ioannis G Kevrekidis and Giovanni Samaey. Equation-free multiscale computation: Algorithms and applications. *Annual review of physical chemistry*, 60:321–344, 2009.

[83] Ian Knowles and Robert J Renka. Methods for numerical differentiation of noisy data. *Electronic Journal of Differential Equations*, pages 235–246, 2014.

[84] J Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. OUP Oxford, 2013.

[85] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*, volume 149. SIAM, 2016.

[86] Kody JH Law and Andrew M Stuart. Evaluating data assimilation algorithms. *Monthly Weather Review*, 140(11):3757–3782, 2012.

[87] Yoonsang Lee and Andrew J Majda. Multiscale methods for data assimilation in turbulent systems. *Multiscale Modeling & Simulation*, 13(2):691–713, 2015.

[88] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam, 2007.

[89] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999.

[90] J.-C. Loiseau and S. L. Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[91] J.-C. Loiseau, B. R. Noack, and S. L. Brunton. Sparse reduced-order modeling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

[92] Jean-Christophe Loiseau and Steven L Brunton. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[93] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668*, 2017.

[94] Richard W Longman and Jer-Nan Juang. Recursive form of the eigensystem realization algorithm for system identification. *Journal of Guidance, Control, and Dynamics*, 12(5):647–652, 1989.

[95] Edward N Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sciences*, 20(2):130–141, 1963.

[96] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[97] Zhixin Lu, Brian R Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.

[98] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[99] Alan Mackey, Hayden Schaeffer, and Stanley Osher. On the compressive spectral method. *Multi. Mod. Sim.*, 12(4):1800–1827, 2014.

[100] Andrew J Majda, John Harlim, and Boris Gershgorin. Mathematical strategies for filtering turbulent dynamical systems. *Discrete Contin. Dyn. Syst*, 27(2):441–486, 2010.

[101] Andrew J Majda, Di Qi, and Themistoklis P Sapsis. Blended particle filters for large-dimensional chaotic dynamical systems. *Proceedings of the National Academy of Sciences*, page 201405675, 2014.

[102] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[103] Niall M Mangan, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, 2016.

[104] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. In *Proc. R. Soc. A*, volume 473, page 20170009. The Royal Society, 2017.

[105] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets: Deep learning of molecular kinetics. *Nature Communications*, 9(5), 2018.

[106] Jerrold E Marsden and Matthew West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

[107] Peter S Maybeck. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982.

[108] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.

[109] Igor Mezic. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.

[110] Srboljub M Mijailovich, Jeffrey J Fredberg, and James P Butler. On the theory of muscle contraction: filament extensibility and the development of isometric force and stiffness. *Biophysical journal*, 71(3):1475–1484, 1996.

[111] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[112] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[113] Bruce Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE transactions on automatic control*, 26(1):17–32, 1981.

[114] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[115] B. R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *J. Fluid Mech.*, 497:335–363, 2003.

[116] Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

[117] Shaowu Pan and Karthik Duraisamy. Data-driven discovery of closure models. *SIAM Journal on Applied Dynamical Systems*, 17(4):2381–2413, 2018.

[118] Shaowu Pan and Karthik Duraisamy. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, 2018, 2018.

[119] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[120] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos*, 27(12):121102, 2017.

[121] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: using machine learning in conjunction with a knowledge-based model. *Chaos*, 28(4):041101, 2018.

[122] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[123] M. Phan, L. G. Horta, J. N. Juang, and R. W. Longman. Linear system identification via an asymptotically stable observer. *Journal of Optimization Theory and Applications*, 79:59–86, 1993.

[124] M. Phan, J. N. Juang, and R. W. Longman. Identification of linear-multivariable systems by identification of observers with assigned real eigenvalues. *The Journal of the Astronautical Sciences*, 40(2):261–279, 1992.

[125] René Pinnau. Model reduction via proper orthogonal decomposition. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 95–109. Springer, 2008.

[126] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *arXiv preprint arXiv:1811.05537*, 2018.

[127] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[128] Patrick Nima Raanes. On the ensemble rauch-tung-striebel smoother and its equivalence to the ensemble kalman smoother. *Quarterly Journal of the Royal Meteorological Society*, 142(696):1259–1264, 2016.

[129] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[130] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[131] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683–693, 2017.

[132] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[133] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[134] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[135] Volker Roth and Bernd Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 848–855. ACM, 2008.

[136] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D.S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 645:115–127, 2009.

[137] Clarence W Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013, 2005.

[138] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

[139] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[140] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *arXiv preprint arXiv:1808.02578*, 2018.

[141] Themistoklis P Sapsis and Andrew J Majda. Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems. *Proceedings of the National Academy of Sciences*, 110(34):13705–13710, 2013.

[142] Simo Särkkä. Unscented rauch–tung–striebel smoother. *IEEE Transactions on Automatic Control*, 53(3):845–849, 2008.

[143] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *PNAS*, 110(17):6634–6639, 2013.

[144] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. In *Proc. R. Soc. A*, volume 473, page 20160446. The Royal Society, 2017.

[145] Hayden Schaeffer and Scott G McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.

[146] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *arXiv preprint arXiv:1707.08528*, 2017.

[147] Hayden Schaeffer, Giang Tran, and Rachel Ward. Learning dynamical systems and bifurcation via group sparsity. *arXiv preprint arXiv:1709.01558*, 2017.

[148] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

[149] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, August 2010.

[150] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[151] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[152] Michael Schmidt, Ravishankar Vallabhajosyula, J. Jenkins, Jonathan Hood, Abhishek Soni, John Wikswo, and Hod Lipson. Automated refinement and inference of analytical models for metabolic networks. *Phys. bio.*, 8(5):055011, 2011.

[153] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[154] Urs Stoecker, Ivo A Telley, Edgar Stüssi, and Jachen Denoth. A multisegmental cross-bridge kinetics model of the myofibril. *Journal of theoretical biology*, 259(4):714–726, 2009.

[155] K. Taira and T. Colonius. The immersed boundary method: a projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007.

[156] K. Taira and T. Colonius. The immersed boundary method: a projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007.

[157] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.

[158] Bertrand CW Tanner, Thomas L Daniel, and Michael Regnier. Sarcomere lattice geometry influences cooperative myosin binding in muscle. *PLoS computational biology*, 3(7):e115, 2007.

[159] Stephal Thaler, Ludger Paehler, and Nikolaus A Adams. Sparse identification of truncation errors. *arXiv preprint arXiv:1904.03669*, 2019.

[160] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. B*, page 267, 1996.

[161] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.

[162] Lloyd N Trefethen. *Spectral methods in MATLAB*, volume 10. Siam, 2000.

[163] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[164] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

[165] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[166] Claudia Veigel, Justin E Molloy, Stephan Schmitz, and John Kendrick-Jones. Load-dependent kinetics of force production by smooth muscle myosin measured with optical tweezers. *Nature cell biology*, 5(11):980, 2003.

[167] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. R. Soc. A*, 474(2213):20170844, 2018.

[168] Katsuzo Wakabayashi, Yasunobu Sugimoto, Hidehiro Tanaka, Yutaka Ueno, Yasunori Takezawa, and Yoshiyuki Amemiya. X-ray diffraction evidence for the extensibility of actin and myosin filaments during muscle contraction. *Biophysical Journal*, 67(6):2422–2435, 1994.

[169] Sam Walcott. Muscle activation described with a differential equation model for large ensembles of locally coupled molecular motors. *Physical Review E*, 90(4):042717, 2014.

[170] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[171] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *PRL*, 106:154101, 2011.

[172] Christoph Wehmeyer and Frank Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of chemical physics*, 148(24):241703, 2018.

[173] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[174] Karen Willcox and Jaime Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA journal*, 40(11):2323–2330, 2002.

[175] C David Williams, Michael Regnier, and Thomas L Daniel. Axial and radial forces of cross-bridges depend on lattice spacing. *PLoS computational biology*, 6(12):e1001018, 2010.

[176] C David Williams, Michael Regnier, and Thomas L Daniel. Elastic energy storage and radial forces in the myofilament lattice depend on sarcomere length. *PLoS computational biology*, 8(11):e1002770, 2012.

[177] C David Williams, Mary K Salcedo, Thomas C Irving, Michael Regnier, and Thomas L Daniel. The length–tension curve in muscle depends on lattice spacing. *Proceedings of the Royal Society B: Biological Sciences*, 280(1766):20130697, 2013.

[178] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[179] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*, 2017.

[180] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics letters A*, 150(5-7):262–268, 1990.

[181] Linan Zhang and Hayden Schaeffer. On the convergence of the sindy algorithm. *arXiv preprint arXiv:1805.06445*, 2018.

[182] Tong Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1921–1928. Curran Associates, Inc., 2009.

[183] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

[184] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.