Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Applications of Kernel-Based Extended Dynamic Mode Decomposition

Christina Nuss-Brill

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Applications of Kernel-Based Extended Dynamic Mode Decomposition

Author:	Christina Nuss-Brill
1st examiner:	Dr. Felix Dietrich
Assistant advisor:	Erik Bolager M.Sc.
Submission Date:	August 13th, 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

August 13th, 2023 Christina Nuss-Brill

Acknowledgments

First of all, I would like to thank Dr. Felix Dietrich and Erik Bolager for the considerable amount of time they spent guiding me through this thesis. I greatly appreciate all the detailed edits and extensive discussions to help me better understand the mathematical concepts. I would also like to thank all of my friends and family for providing me with support as I worked through this chapter of my life. Finally, I would also like to thank TU Munich as a whole for providing me with the great opportunity to study in Germany and achieve a master's degree.

"When you stop doing things for fun, you might as well be dead."

- Ernest Hemingway

Abstract

Extended Dynamic Mode Decomposition (EDMD) is a powerful data-driven numerical method based on the theoretical concepts of Koopman operator theory. It can be used to approximate and predict the behavior of a dynamical system using sample data without requiring any knowledge of the underlying dynamics. This is particularly useful for predicting the dynamics of nonlinear systems by summarizing them into a linear operator, which acts on a space of observable functions to create a reduced-order model of the original system. Kernel EDMD is an extension of EDMD that uses a kernel function to efficiently utilize a high-dimensional dictionary of observables. The theory of Kernel EDMD is well established; however, the accuracy of its practical implementation is highly dependent on the choice of functions in the dictionary. This thesis details a method for finding an optimal choice of dictionary functions through the use of kernel parameter learning with a gradient method called kernel EDMD-DL. This method is applied to numerically generated data from two different dynamical systems, a hopf bifurcation and a hydrogen combustion reaction in a constant volume transient CSTR. The data is successfully reconstructed for both systems, and the accuracy is reported along with an overview of the techniques used for optimal hyperparameter tuning and data cleanup, which are critical components to the success of the method. A strong emphasis is made on using the kernel EDMDL-DL method for making out-of-sample predictions. The knowledge gained through this work can be used to enhance an understanding of the capabilities of kernel EDMDL-ML and how it can be applied to complex nonlinear systems existing in engineering and science applications.

Contents

Acknowledgements **vii**

Abstract **ix**

1 Introduction **1**

2 State of the Art **3**

2.1 Dynamical Systems and the Koopman Operator 3

2.1.1 Linear Approximation of Nonlinear Systems 3

2.1.2 Dynamic Systems 4

2.1.3 Defining the Koopman Operator 5

2.1.4 Linear Decomposition of the Koopman Operator 6

2.2 Data Driven Methods for Koopman Approximation 7

2.2.1 DMD Method 8

2.2.2 EDMD Method 11

2.2.3 Using Kernels 14

2.2.4 Kernel Basics 14

2.2.5 Eigenfunctions, Eigenvectors, and Modes 15

2.2.6 Kernel EDMD-DL 16

2.2.7 Recreating the System Dynamics 17

3 Applications of Kernel EDMD **19**

3.1 Experimental Methodology 19

3.1.1 Data Collection 19

3.1.2 Reporting Accuracy 20

3.1.3 Kernel Selection 21

3.1.4 Hyperparamter Optimization 22

3.1.5 Eigenfunctions and Eigenvalues 24

3.2 Application of Kernel EDMD to Hopf Bifurcation 25

3.2.1 Hopf System Background Theory and Data Collection 25

3.2.2 Hopf System Results 26

3.2.3 Hopf System Hyperparameter Optimization 27

3.2.4 Learning the Parameterized RBF Kernel Using Kernel EDMD-DL . . 28

3.3 Application of Kernel EDMD to Hydrogen Combustion CSTR 30

3.3.1 Hydrogen Combustion Background Theory 31

3.3.2 CSTR Reactor Background Theory 33

3.3.3 Defining and Solving the Dynamic Reaction System 33
3.3.4 Simultaneous Modeling of Temperature and Hydrogen Concentration Variations 38
3.3.5 Separately Modeling Temperature and Hydrogen Concentration Variations 44
3.3.6 Modeling a Second Dynamic Reaction System 48

4 Conclusion **53**
4.1 Summary 53
4.2 Discussion and Outlook 53

Bibliography **55**

1 Introduction

In modern society, numerical modeling methods go hand in hand with classical engineering and science disciplines. Understanding and predicting the behavior of systems that evolve over time is critical to any application in these fields. The ability to model such dynamical systems is very useful because predictions with numerical simulations can enhance understanding of the system beyond the bounds of experimental testing to accelerate research and development. Physical systems encountered in real-world situations are extremely complex, and most often, the mathematical equations governing the system's evolution are unknown. These dynamic systems can be extremely high dimensional and it can be the case that not all of the variables contributing to the system's evolution are even identified. These aspects can cause difficulty when one wants to use numerical modeling.

Currently, many machine learning techniques have been designed to take advantage of the current technological advances to collect and process high quantities of data in an attempt to model such complex systems, for example, physics-informed neural networks [25] and time-lagged autoencoders [30]. Another such method, called Extended Dynamic Mode Decomposition (EDMD), is a comparatively efficient model that requires less data than a Neural network [1]. It is also a completely data-driven numerical method used for modeling dynamic systems. EDMD is able to recreate the dynamics of an underlying system only from sampled data points of the evolving system with no knowledge of the underlying dynamical system governing equations. The EDMD method uses sampled data to calculate a linear operator, which acts on a space of observable functions to create a reduced-order approximation of the original system. Utilizing linear operators to approximate nonlinear dynamics holds the potential to significantly improve prediction and control capabilities such complex systems [5].

EDMD is heavily based on the mathematics behind Dynamic Mode Decomposition (DMD) [27], and both methods are intimately connected with Koopman operator theory [17]. This theory is based on the existence of an infinite dimensional linear operator that can recreate the dynamics of a nonlinear dynamic system by trading nonlinearity for infinite dimensionality. Both DMD and EDMD calculate a finite-dimensional approximation for the Koopman operator. This operator can be used to recreate the underlying dynamical system and predict the dynamics of the system outside of the data used to create it.

EDMD extends DMD in the sense that it allows for the incorporation of a broader range of basis functions from which to combine and approximate the dynamic system [21]. It maps the original data into a higher-dimensional feature space where the dynamics of the system can be approximated linearly. While the theories of EDMD are well established, the practical implementation is very difficult because the accuracy of the reconstruction of the dynamic system is highly dependent upon the choice of basis functions. The appropriate choice of basis functions can be unique to the underlying system and it is nearly impossible for one to choose the right basis functions without any knowledge of the mathematical construction of the underlying system.

The difficulty in choosing an appropriate set of basis functions is the motivation behind kernel EDMD which augments EDMD with a kernel function that can represent a high dimensional space of basis functions for a very low computational cost using the kernel trick. There are many different types of kernel functions and each one represents a different dictionary of basis functions. Some kernels have free parameters that can also affect the characteristics of the functions represented by the kernel.

This thesis describes a method called Kernel EDMD-ML, which advances the concept of kernel EDMD [3]. It employs the use of machine learning gradient methods to optimize the free parameter in parameterized kernels. Essentially, this assists in finding an optimized set of basis functions that best represent the underlying dynamic system.

The main goal of this thesis is to apply Kernel EDMD to two nonlinear dynamical systems with numerically generated data samples. The implementations are discussed in detail, and a report is made on the accuracy of using the approximated linear operators to reconstruct the original sample data and predictions outside the original data range. The two systems used are the cubic Lotka-Volterra equations with a Hopf bifurcation and a bifurcation created by a dynamic system representing the dynamics of hydrogen combustion in a constant volume transient Continuous stirred tank reactor (CSTR) [20][7]. The first dynamic system is well-studied and is a classic example of a system characterized by rather simple nonlinear dynamics; thus, it is used as a proof of concept for both Kernel EDMD and Kernel EDMD-DL. For the second system, only Kernel EDMD-MLis be applied. This is a much more complicated and higher-dimensional system. It provides an excellent illustration of how Kernel EDMD-DL could potentially be employed to model real-world experimental data.

The work of this thesis is split into three parts, the first part describes in detail the mathematical background behind Koopman operator theory, DMD, EDMD, kernel EDMD, and kernel EDMD-ML. Then, in the second part, the kernel EDMD methods are applied to the two dynamical systems. The hyperparameters are tuned and the most optimal implementations of the methods are presented. These results are used to assess the efficacy of the methods. Particular emphasis is placed on evaluating the precision of predictions for the dynamic system beyond the scope of the sample data, as this highlights a critical component of the method's applicability to model real-world dynamic systems.

2 State of the Art

2.1 Dynamical Systems and the Koopman Operator

As a big picture, Koopman Operator theory is a relatively modern approach to analyzing nonlinear dynamic systems. Analysis of differential equations is essential in many fields of science and engineering, often in the context of prediction, control, and optimization. The first idea of the Koopman operator traces back to Bernard O. Koopman in 1931, who presented the theory of an infinite dimensional linear operator which can recreate the dynamics of a nonlinear dynamic system trading nonlinearity for infinite dimensionality [17]. It provides a global understanding of the evolution of the system through one time iteration. Initially, the Koopman Operator was directly connected to Ergodic theory but has since been generalized and expanded upon in later works such as by the works of Mezic et. al. [19]. The real application of the Koopman Operator is a data-driven approach, and the accuracy is related to the amount of data used for analysis. As such, Koopman theory has seen a surge in interest with modern computing power. As such, Koopman theory has seen a surge in interest with modern computing power. The capacity to linearly represent a nonlinear system enhances predictive capabilities by offering the opportunity to utilize spectral analysis techniques effectively.

2.1.1 Linear Approximation of Nonlinear Systems

A system is defined as nonlinear when the change in output is not proportional to a change in the input. This happens when the system does not exhibit additivity and homogeneity both of which are required for the output to be a linear combination of the input variables. Real-world problems are often nonlinear. These nonlinear dynamical systems are not always analytically solvable and can exhibit complex behavior such as periodic variation, limit cycles, and bifurcations. They also contain more than one steady state which limits the applicability of classical eigendecomposition [26]. Additionally, nonlinear systems are sensitive to noisy data and small changes in the initial conditions. The usual approach to analyzing a nonlinear system is to understand the geometry of families of solutions in phase space as well as local linearizations near stability points [6][26].

Because the Koopman operator is a linear operator, the nonlinear system it represents can be analyzed using traditional eigendecomposition techniques instead of geometric techniques [6]. Spectral decomposition of the infinite dimensional Koopman operator completely characterizes the nonlinear system. For some simple select nonlinear systems, the system can be represented by closed Koopman invariant subspaces, however, the majority of functions cannot be [5].

2.1.2 Dynamic Systems

Underlying Koopman Operator theory is the theory of dynamical systems. A dynamical system describes how a point within a set evolves globally over time. This system is composed of a triple including a phase space $\mathcal{M}$, a rule that specifies how a point within the phase space evolves with time ϕ, and a set of times T [16]. The rule is defined as

$$\phi : \mathcal{M} \times T \rightarrow \mathcal{M} \tag{2.1}$$

with a continuous-time dynamical system $T \subseteq \mathbb{R}$ and for a discrete dynamical system $T \subseteq \mathbb{Z}$.

Koopman theory considers a specific way to represent the behavior of a dynamic system. In literature, the Koopman operator has been applied to many different types of discrete and continuous dynamical systems. Abstractly, the Koopman operator exists for any forward mapping; however, for the operator to be approximated, certain requirements of the state space $\mathcal{M}$ (in the context of Koopman analysis, one generally considers $\mathcal{M}$ to be a state space and not a full phase space) and mapping ϕ should be met. Some of these are that the underlying state space should be finite-dimensional, compact, and observable or measurable in some way [21]. It is also often assumed that the mapping is autonomous, continuous, and deterministic. Most of the time, the dynamic system is defined over a measure space with a measure-preserving map, but this is not necessary [6][8].

In this thesis, as in most cases in the literature, the Koopman operator is applied to dynamic systems governed by a system of ordinary differential equations (ODEs). The ODEs are continuous-time dynamical systems accompanied by appropriate boundary and initial conditions with Lipschitz continuity. The evolution operator acts on states in the state space. The states x are defined by a vector field in a Euclidean space $x \in \mathcal{M} \subseteq \mathbb{R}^n$, $(t, x(t)) \in U$ and $U \subseteq T \times \mathbb{R}^n$, and $T \subset \mathbb{R}$. $f : U \rightarrow \mathbb{R}^n$ is a continuous function that is sufficiently differentiable. The initial conditions are also defined $(t_0, x_0) \in U$ [16].

For the sake of simplicity, the ODEs considered are of the first order and autonomous. ODEs of higher order can always be transformed into a system of first-order ODEs. Autonomous means the t argument is not present on the right hand side. These conditions result in an ODE of the form

$$\frac{d}{dt}x(t) = f(x(t)). \tag{2.2}$$

Although linear and nonlinear ODEs are applicable to Koopman analysis, nonlinear ODEs are the most interesting because many techniques already exist for the analysis and solution of linear ODEs. A non-linear first-order ODE is an ODE where $f(x(t))$ involves nonlinear operations in $x(t)$ (i.e. exponentiation). Koopman analysis can also be applied to systems of ODEs. A system of ODEs is a set of two or more interconnected differential equations involving multiple unknown functions and their derivatives with respect to the same independent variable.

2.1.3 Defining the Koopman Operator

The calculation of the Koopman operator is a data driven process so it is most natural to consider discrete-time dynamic systems which are [21]

$$x_{n+1} = F(x_n), \quad x(n) \in \mathbb{R}^N, \quad n \geq 0 \quad n \in \mathbb{N}. \tag{2.3}$$

Discrete-time dynamic systems can be induced from continuous ones by sampling a continuous time system at discrete time points [5], so the discrete-time evolution operator is parameterized by the time step Δt. Starting with the autonomous first ODE as defined in 2.1.2, the discrete system can be related to the continuous one through

$$F(x(t_0)) = x(t_0) + \int_{t_0}^{t_0+\Delta t} f(x(\tau))d\tau. \tag{2.4}$$

A critical characteristic of the Koopman operator is that it does not act directly on the states in the state space but rather describes how the scalar measurement functions (observables) defined on the state space evolve [21]. The observables are defined as

$$g : \mathcal{M} \rightarrow \mathbb{R}, \tag{2.5}$$

which can be combined to create the vector-valued observable $\mathbf{g} : \mathcal{M} \rightarrow \mathbb{R}^M$

$$\mathbf{g}(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \cdot \\ \cdot \\ \cdot \\ g_M(x) \end{bmatrix}, \tag{2.6}$$

where g_n belongs to some function space $\mathcal{F}$. Usually, these functions belong to the Hilbert space $\mathcal{F} = \mathcal{L}_2(\mathcal{M}, \rho)$, which is endowed with its corresponding inner product and norm [21]. In this thesis, we will only consider a mapping to the real numbers $\mathbb{R} \subset \mathbb{C}$.

The Koopman operator $\mathcal{K}$ describes the linear evolution of $\mathbf{g}$ as governed by the underlying dynamical system

$$\mathcal{K}\mathbf{g}(x_n) = \mathbf{g}(F(x_n)) = \mathbf{g}(x_{n+1}), \tag{2.7}$$

where $F(x(n))$ is the discrete evolution operator of the dynamic system. The intent of the Koopman operator is not to map states to states but to map functions over a state space to functions over a state space. The Koopman operator controls the change of the observable functions over time [31]. Equation 2.7 works for all observable functions over all states $x \in \mathcal{M}$. The observable functions do not have to be linearly independent.

It is also possible to define the Koopman operator for continuous time dynamical systems directly instead of first having to discretize the underlying governing equation. This however will not be discussed as it is not a topic of focus for this thesis.

2.1.4 Linear Decomposition of the Koopman Operator

Because the Koopman operator is linear, it can be decomposed into its eigenvalues and eigenvectors using traditional eigendecomposition techniques. The concept is the same for both the discrete and continuous versions of the Koopman operator. The decomposition can then be used to construct eigenfunctions of the operator which provide a linear summary of its action on the observables. An observable is an eigenfunction of the Koopman operator $\mathcal{K}$ if

$$\varphi(x_{n+1}) = \mathcal{K}\varphi(x_n) = \lambda\varphi(x_n). \tag{2.8}$$

The eigenfunctions of the Koopman operator are represented by $\varphi : \mathcal{M} \to \mathbb{C}$ and the eigenvalues by $\lambda \in \mathbb{C}$. The set of eigenfunctions of the Koopman operator does not necessarily form a complete basis for the function space of observables. The observables can be rewritten as the projection onto the span of the eigenfunctions as in [6]

$$g_i(x) = \sum_{j=1}^{\infty} \varphi_j(x) v_{ij}. \tag{2.9}$$

.

Expanding 2.10 to the vector of all observable functions, a vector of coefficients $\mathbf{v}_j(\psi) \in \mathbb{C}^N$ called the Koopman Modes can be uncovered by using

$$\mathbf{g}(x) = \sum_{j=1}^{\infty} \varphi_j(x)\mathbf{v}_j. \tag{2.10}$$

The eigenvalues λ represent the time-dependent behavior of the system, the eigenfunctions are determined by the initial conditions $\varphi(x)$, and the Koopman modes are the projection of the observables onto the space spanned by the eigenfunctions $\mathbf{v}_j(x)$ [6]. With this in mind, the original states of the system can be reconstructed by a tuple called Koopman mode decomposition [5], which is

$$\mathbf{g}(x_n) = \mathcal{K}^n\mathbf{g}(x_0) = \sum_{j=0}^{\infty} \lambda_j^n \varphi_j(x_0)\mathbf{v}_j. \tag{2.11}$$

Figure 3.2 shows a quick summary of the Koopman operator theory as it is covered in section 2.1 taken from [31]. It describes graphically how the Koopman Eigenfunctions and Koopman Modes relate to the underlying dynamical system.

A final topic to cover is the Koopman invariant subspace. This is a space composed of the span of all observable functions

$$g = a_1 g_1 + a_2 g_2 ... + a_p g_p, \tag{2.12}$$

of which remain in the subspace after being acted upon by the Koopman operator through

$$\mathcal{K}g = \beta_1 g_1 + \beta_2 g_2 ... + \beta_p g_p. \tag{2.13}$$

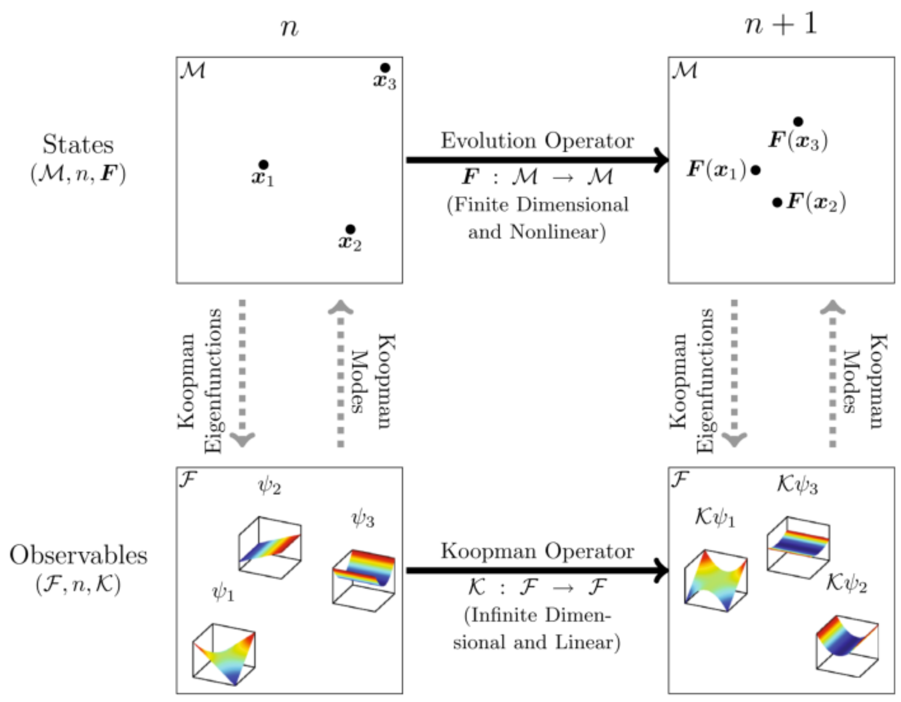

n
$n+1$
States
$(\mathcal{M}, n, \boldsymbol{F})$
$\mathcal{M}$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}_3$
Evolution Operator
$\boldsymbol{F} : \mathcal{M} \to \mathcal{M}$
(Finite Dimensional and Nonlinear)
$\boldsymbol{F}(\boldsymbol{x}_1)$
$\boldsymbol{F}(\boldsymbol{x}_2)$
$\boldsymbol{F}(\boldsymbol{x}_3)$
Koopman Eigenfunctions
Koopman Modes
Observables
$(\mathcal{F}, n, \mathcal{K})$
$\mathcal{F}$
ψ_1
ψ_2
ψ_3
Koopman Operator
$\mathcal{K} : \mathcal{F} \to \mathcal{F}$
(Infinite Dimensional and Linear)
$\mathcal{K}\psi_1$
$\mathcal{K}\psi_2$
$\mathcal{K}\psi_3$

Figure 2.1: Illustration of Koopman Operator Theory taken from [31].

Any set of the eigenfunctions of the Koopman operator will span an invariant subspace [4]. If one can find a subset of equations that are closed under the dynamics operator, then a finite number of observable functions is possible as well as a finite-dimensional linear Koopman operator. For many systems, however, it may still be that an infinite number of functions within the invariant subspace are required.

2.2 Data Driven Methods for Koopman Approximation

To calculate a finite approximation of the Koopman operator, the governing equations for the underlying dynamics of the system do not need to be known. Rather, the Koopman operator can be approximated from either experimental or numerically calculated samples of the system. These are called data driven methods. These methods focus on finding a finite dimensional representation of the Koopman operator. In this case, a subset of the Koopman invariant subspace is searched for which captures the greatest variation in the dynamics. Some common data driven methods are DMD [27], Generalized Laplace analysis [6], the Ulam Galerkin method [9], Space identification of nonlinear dynamics (sINDy) [5], and Extended Dynamic Mode Decomposition (EDMD) [31]. Data-driven methods generate a finite dimensional linear approximation of the Koopman operator with spectral properties which are similar to those of the true operator [21]. The Kernel EDMD-DL method described in this thesis leverages EDMD which is heavily based on the traditional DMD method. The next sections will first review the critical components of the traditional DMD, then discuss how the EDMD method expands upon DMD.

2.2.1 DMD Method

EDMD gets its origins from the original DMD method which is likely the most popular method due to its numerical stability and simplicity. DMD unrelated to the Koopman operator, finds an approximate linear system that advances high dimensional measurements of a state space in time. It also does not require any understanding of the underlying dynamical system and instead relies purely on data [5]. If the DMD method is applied to finding a finite dimensional approximation of the Koopman operator, the final result of the algorithm is the group of eigenfunctions and eigenvalues of the Koopman operator which can be, in the end, combined to recreate the dynamics of the system. There are many different versions of DMD. In this section, we go into more detail about the DMD algorithm based on the work in [28] which utilizes singular value decomposition.

To begin DMD, data is recorded from a dynamical system with a state space with P dimensions. This data can either be experimental or numerically generated. Typically the data is collected in the form of a discrete time-series of the data with $N \in \mathbb{Z}$ time points. Values of the state space are recorded at discretized time intervals from multiple discretized locations in the state space [5]. Next, the states are sampled from the data as discrete-time snapshot pairs denoted as $\{x(t_n), x(t_{n+1})\}_{n=1}^{N}$. One snapshot pair includes the states vector $x(t_n)$ at time t_n, and its partner is the states vector at the next snapshot in the time series with the movement of the states governed by the discrete differential operator $y_n = y(t_n) = x_{n+1} = x(t_{n+1}) = F(x(t_n)) = F(x_n)$. The sampling times of the pairs can be either irregular or uniform and could belong to combined data from several different experiments [4]. If uniform sampling is used, after a period of time $t_n = n\Delta t$.

Finally, all of the sampled x state vectors in the time series are stacked into a long $P \times N$ matrix X, and all the sampled y state vectors are stacked into a long $z \times N$ matrix Y. This results in two matrices with the columns of Y right shifted from X by one discrete time period Δt. In theory, the data does not necessarily have to come from a time series for the DMD algorithm to work but rather X and Y must just be linearly consistent [28].

$$X = \begin{bmatrix} | & | & | & | \\ x_1 & x_2 & \dots & x_{N-1} \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{bmatrix}$$
$$Y = \begin{bmatrix} | & | & | & | \\ x_2 & x_3 & \dots & x_N \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{y}_4 \end{bmatrix} \tag{2.14}$$

As an example, imagine a physical dynamic system that models the trajectory of a moving particle in 2D space ($P = 2$). The particles are placed at Z initial discretized locations, and movement of the system begins a $t = 0$. From $t = 0$ to $t = N$ the spacial locations of each trajectory are recorded. In the end, there are $N \times 2 \times Z$ numerical data points. The data is then separated into Z separate matrices of size $2 \times N$ with each matrix containing the state vectors from one trajectory. Next, Z separate X and Y matrices are created each with a total of $N - 1$ columns. Finally, the entire data from the Z separate trajectories is put together by stacking the X matrices from each trajectory columnwise and stacking the Y matrices from each trajectory columnwise. This results in large X and Y matrices of size $Z \times N$. For the rest of the introduction section for ease of notation, it is assumed data is collected from a system similar to the scenario of only one "trajectory" so X and Y have sizes of $P \times N$.

Next, for DMD we assume a linear correlation between X and Y with an operator A which linearly advances the system between the snapshots.

$$Y \approx AX \tag{2.15}$$

To estimate the operator A, a least squares approximation is used. The best-fit operator A is calculated using the pseudo-inverse of X which is defined by the minimization problem

$$A = \arg\min_{A} J(A) = \arg\min_{A} ||Y - AX||_F = YX^{+}. \tag{2.16}$$

with F indicating the Frobenius norm and X^{+} indicating the psuedo-inverse of X. Going back to the definition of the Koopman operator in 2.7, if one considers the Koopman operator observation functions to be linear measurements of the states $g(x_n) = x_n$, then A can be considered a finite-dimensional approximation for the Koopman operator [4].

The matrix A is square, the size of which is determined by the size of the states vectors. DMD is often applied to systems with high dimensional state vectors (such as for turbulent fluid flow over an airfoil), so A is likely to be very large. The DMD algorithm avoids explicit computations of A directly by using the pseudo-inverse of X to perform dimensionality reduction by computation of the eigenvalues and eigenvectors of A. The pseudo inverse is calculated via Singular Value Decomposition (SVD). Using SVD makes the algorithm more efficient because it computes the projection of A onto the leading singular vectors of X.

SVD factors a matrix X into three separate matrices, which provide more useful information about how the original matrix acts in a linear transformation. It is a generalization of the eigendecomposition of square matrices to any rectangular matrix.

$$X \approx \mathrm{Q}\Sigma \mathrm{Z}^{T} \tag{2.17}$$

If P is equal to the dimension of the state space and N is equal to the number of snapshots then $Q \in \mathbf{R}^{P\times P}$, $Z \in \mathbf{R}^{N\times N}$, and $\Sigma \in \mathbf{R}^{P\times N}$. Where both Q and Z are unitary matrices. DMD can also be used for complex matrices but in this thesis only real matrices will be considered. When X is decomposed, A can be calculated as follows:

$$A = YZ\Sigma^{-1}\mathrm{Q}^T \tag{2.18}$$

The columns of Q (left singular vectors) and Z (right singular vectors) are orthonormal and the matrix Σ is a diagonal matrix with positive real entries which are the singular values of X. Q and Z are calculated from the correlation matrices XX^T and X^TX which provide information about the how each variable inside X is linearly related to the others row-wise and column-wise. The columns of Z consist of the transpose of the eigenvectors of X^TX and the columns of Q consist of the eigenvectors of XX^T. The right and left eigenvectors of X can be related to the correlation matrices through

$$X^TX = Z\Sigma Q^T Q\Sigma Z^T = Z\Sigma^2 V^T = XX^TV = \Sigma^2 V^T \tag{2.19}$$
$$XX^T = Q\Sigma Z^T Z\Sigma Q^T = Q\Sigma^2 Q^T = XX^TQ = Q\Sigma^2. \tag{2.20}$$

The matrix Σ contains the square roots of the eigenvalues from X^TX arranged in descending order. The number of non-zero singular values in Σ will be at most as many as the size of the largest dimension of X. For example, if $P > N$, Σ will have at most N non-zero singular values. This results in the computed A matrix being at most $N \times N$. It is possible to truncate Σ and keep only the first r leading singular values. This reduces Σ to size $\mathbf{R}^{r\times r}$. This also requires truncating Q and Z accordingly to $Q \in \mathbf{R}^{P\times r}$, $Z \in \mathbf{R}^{N\times r}$. If this reduction occurs, A now becomes $r \times r$. The eigenspace of the truncated A can only be shown to be an approximation of the true eigenspace. It is important to note that Q is no longer unitary, and the solution to the least squares approximation will be a projection onto the column space of Q [5].

After truncation, A can now be approximated by projection of A onto the column space of Q to reduce the dimensionality [28].

$$\tilde{A} = \mathrm{Q}^T A\mathrm{Q} = \mathrm{Q}^T YZ\Sigma^{-1}. \tag{2.21}$$

The orthogonal columns of Q are important and are known as the Proper Orthogonal Decomposition (POD) modes. The POD modes are temporally averaged spatial auto-covariance matrix of which the eigenvectors are computed and arranged as columns ranked according to their energy content by Σ [5].

Next, the left eigenvalues and eigenvectors of $\tilde{A}$ are calculated using the usual eigendecomposition

$$\tilde{A}\mathrm{W} = \Lambda\mathrm{W}. \tag{2.22}$$

If Σ is only truncated up to the non-zero singular values the eigenvalues of $\tilde{A}$ are the same as the full A matrix. Finally, the DMD modes can be calculated from the eigenvectors of $\tilde{A}$ by transforming them back to the column space of Y

$$\Phi_{DMD} = YZ\Sigma^{-1}\mathrm{W}. \tag{2.23}$$

Through some algebra, the DMD modes can be shown to also be the eigenvectors of the original A matrix corresponding to the eigenvalues Λ

$$A\Phi_{DMD} = \Phi_{DMD}\Lambda \tag{2.24}$$

Finally, the original states can be reconstructed, and the dynamic system can be now be represented as a spectral decomposition

$$x_n = \Lambda^{n-1}\Phi\mathbf{b}. \tag{2.25}$$

Equation 2.25 is comparable to equation 2.10 when defining how to reconstruct the original system using the Koopman operator. $\mathbf{b}$ is a vector which represents the mode amplitudes. It can be estimated by solving $\mathbf{x}_1 = \Phi\mathbf{b}$ and $\mathbf{b} = (W\Lambda)^{-1}x_1$ [5]. A summary of the DMD process is wonderfully summarized in figure 2.2.

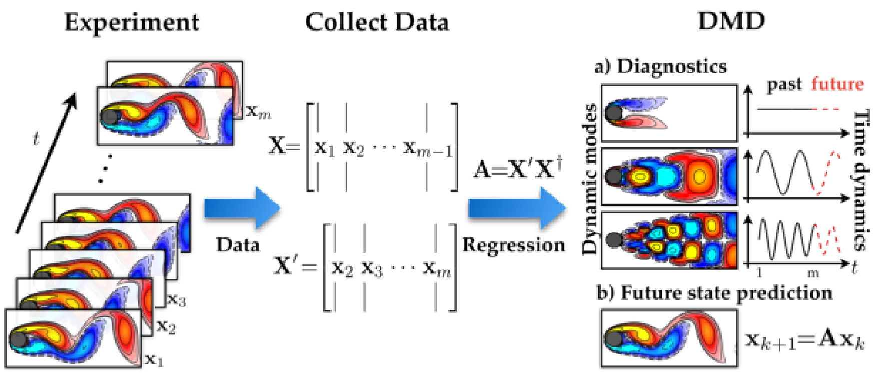

Experiment
Collect Data
DMD
t
$\mathbf{x}_m$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_1$
Data
$\mathbf{X}=[\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_{m-1}]$
$\mathbf{X}'=[\mathbf{x}_2\ \mathbf{x}_3 \cdots \mathbf{x}_m]$
$\mathbf{A}=\mathbf{X}'\mathbf{X}^\dagger$
Regression
a) Diagnostics
past future
Dynamic modes
Time dynamics
1
m
t
b) Future state prediction
$\mathbf{x}_{k+1}=\mathbf{A}\mathbf{x}_k$

Figure 2.2: Holistic Summary of the DMD method illustrated on flow past a cylinder at Reynolds number 100 reproduced from [15].

2.2.2 EDMD Method

In this section, Extended Dynamic Mode Decomposition (EDMD) is discussed by building upon the definition of DMD. As a recap, when considered in the context of the Koopman operator, the DMD algorithm assumes a linear correlation between direct measurements of the state, which equates to linear observable functions $g_i(x_n) = x_n$ from equation 2.5. This often results in an overly simplistic approximation of the Koopman operator for complex non-linear systems. EDMD enhances DMD to consider instead an expanded set of non-linear measurements of the state (this set can also include linear measurements of the states).

To perform EDMD, first, a dictionary of scalar functions is defined as

$$\mathcal{D} = \{\psi_1, \psi_2....\psi_M\}, \tag{2.26}$$

where

$$\psi_i : \mathcal{M} \to \mathbb{R} \quad \text{for} \quad i = 1, 2, ..., M.$$

The span of the functions ψ_i in the dictionary creates a subspace $\mathcal{F}_M$ which belongs to a larger function subspace $\mathcal{F}_M \subset \mathcal{F}$. As stated in section 2.1.3, the space $\mathcal{F}$ belongs to the $\mathcal{L}^2$ space. The dictionary functions can be written in terms of a vector-valued function $\Psi(x) : \mathcal{M} \to \mathbb{R}^M$ as in

$$\Psi(x) = [\psi_1(x), \psi_2(x)....\psi_M(x)]^T. \tag{2.27}$$

Additionally, the dictionary functions can be superimposed with some vector a as

$$g(x) = \sum_{m=1}^{M} a_m \psi_m(x) = \Psi(x)\mathbf{a}. \tag{2.28}$$

When the chosen dictionary is not a Koopman invariant subspace, $\hat{g}(x) = \mathcal{K}g(x) + r(x)$ with a residual $r(x)$, the goal of EDMD is to define a mapping from a to $\hat{a}$ that minimizes the residual of the superimposed dictionary functions at the points x to the underlying dynamic system at the next time point. This linear mapping can be done through a finite-dimensional approximation of the Koopman operator $K \in M \times M$ [32]. As the number of linearly independent basis functions in the dictionary increases, the Koopman approximation converges to the true Koopman operator [19].

To perform EDMD, data is collected in the same way as for DMD in snapshot pairs with data samples of the state space collected in states vectors from $N \in \mathbb{Z}$ discrete time points. These samples are then organized into matrices X and Y. Equation 2.27 can then be applied to the states vectors for each snapshot as in

$$\Psi(X) = [\Psi(x_1), \Psi(x_2)....\Psi(x_{N-1})], \tag{2.29}$$

where $\Psi(X) : \mathcal{M} \to \mathbb{R}^{M \times N}$. $\Psi(Y)$ can also be created when equation 2.27 is applied to all columns of Y.

To calculate an approximation of the operator, a linear least squares approach is taken as in traditional DMD

$$K = \arg\min_{\tilde{K}} J(\tilde{K}) = \arg\min_{\tilde{K}} ||\Psi(Y) - \tilde{K}\Psi(X)||_F^2. \tag{2.30}$$

The solution to equation 2.30 is

$$K = \Psi(Y)\Psi(X)^+. \tag{2.31}$$

After calculating the approximated Koopman operator K, the vector of eigenfunctions Φ of K can be approximated by combining the vector of dictionary functions and the corresponding left w_k eigenvectors of K using

$$\Phi = \text{W}\Psi. \tag{2.32}$$

where W is the matrix created with each row a left eigenvector.

In the traditional EDMD method, the functions in the dictionary are defined explicitly. The important thing is that the functions within the dictionary are different than just a linear function of the states themselves which makes EDMD unique from DMD. The focus of this thesis is to create this extended function space with a very large number of dictionary functions, much greater than the number of snapshots $M >> N$. This is similar to the situation when applying DMD to a state space with higher dimensionality than the number of snapshots. Because of this, we take the same approach as in DMD and use SVD to avoid calculation complexity dependent upon the size of the dictionary.

Next, SVD is used to decompose $\Psi(X)$ through

$$\Psi(X) \approx \hat{\text{Q}}\hat{\Sigma}\hat{\text{Z}}^T. \tag{2.33}$$

Then, because the number of snapshots is assumed to be greater than the number of dictionary functions, Σ sigma can be reduced to $N \times N$. Therefore, Q becomes $M \times N$ and Z stays the same. Q is no longer unitary but semi-unitary. Equation 2.33 can be rewritten as

$$\Psi(X) \approx \text{Q}\Sigma\text{Z}^T. \tag{2.34}$$

Substituting equation 2.33 in 2.31 to solve for K $\hat{K}$ can be approximated by

$$\hat{K} = \Psi(Y)\text{Z}\Sigma^{-1}\text{Q}^T. \tag{2.35}$$

As mentioned in equation 2.21, $\hat{K}$ can now become a projection of K onto the columns of Q due to the truncation of Σ to reduce the dimensionality of $\hat{K}$. A projection operator $P = QQ^T$ can be applied so that $\hat{K}$ becomes K_p through the relation

$$K_P = P\hat{K} = Q\hat{K}Q^T = Q^T\Psi(Y)\text{Z}\Sigma^{-1}. \tag{2.36}$$

Where P indicates the projection operator.

$\hat{K}$ can be shown to have an equivalent eigenspace to K_P, but the eigenspace of K_P is only a subset of the true Koopman operator K eigenspace unless K is somehow invariant on the eigenspace of K_P (Kernel EDMD learning). The left and right eigenvectors of the projected operator K_P can also be related back to the eigenvectors of $\hat{K}$ using Q and Q^T as a map between the two spaces. Assume v and w are right and left eigenvectors of K_P while $\hat{v}$ and $\hat{w}$ are eigenvectors of $\hat{K}$ then

$$w_j = \hat{w}^T Q \tag{2.37}$$

and

$$v_j = Q\hat{v}_j. \tag{2.38}$$

The minimization objective can be rewritten to solve for K_P as

$$K_P = \arg\min_{\tilde{K}} ||P\Psi(Y) - (P\tilde{K})P\Psi(X)||_F^2, \tag{2.39}$$

where $\tilde{K}$ is an approximation for the true Koopman operator. Equation 2.40 has solution $K_P = (P\Psi(Y))\Psi(X))^+$.

2.2.3 Using Kernels

This next section summarizes the work of Williams et. al. [32], who developed the idea of Kernel EDMD. In the SVD decomposition of $\Psi(X)$, the matrix Q is size $M \times N$, which is still very large, and its calculation requires explicit evaluations of the dictionary functions. In equation 2.35, Q can be replaced with the relation $Q = (P\Psi(X))Z\Sigma^+$ from equation 2.34 and through some algebraic manipulation, one can then rewrite $\hat{K}$ as

$$\hat{K} = (Z\Sigma^+)^T(((P\Psi(X)^T)(P\Psi(Y)))(Z\Sigma^+). \tag{2.40}$$

With $\hat{A} = (P\Psi(X)^T)(P\Psi(Y))$ being the gram matrix.

In equation 2.40, The values of Z and Σ are also needed to calculate K_P. This can be done through the method of snapshots which makes use of relationship from equation 2.20 which defines Z as the left eigenvectors of the correlation matrix X^TX as

$$\hat{G} = ((P\Psi(X)^T)(P\Psi(X)) = Z\Sigma^2 Z^T, \tag{2.41}$$

where Σ and Z are the same as in equation 2.34.

The computation of $\hat{A}$ and $\hat{G}$ requires the evaluation of the entire dictionary for every combination of X and Y points in the state space. This requires explicit knowledge of the dictionary and computational time on the order of $O(N^2M)$. A technique called the Kernel trick can be used to replace the function evaluations. When the dictionary is very large, this trick reduces the computational effort and avoids evaluating the functions in the dictionary.

2.2.4 Kernel Basics

Instead of explicitly defining all of the functions in the dictionary, a single function called a kernel function is used. The choice of the kernel function defines the functions in the dictionary in an implicit way. A kernel function is a function $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ which, instead of requiring the evaluation of the functions at X and Y, requires only the inner product between X and Y. For the inner products of functions, the complexity of calculating a combination of inner products becomes dependent upon the size of the states space vector P rather than the size of the function space M over which the states are evaluated [32] because

$$f(x_i, x_j) = \langle x_i, x_j \rangle = \psi(x_i)^T\psi(x_j). \tag{2.42}$$

In the case of EDMD, the kernel function can be used for the least squares estimation of $\hat{K}$ to avoid explicit evaluations of the dictionary functions and to reduce the complexity of computation when the dictionary is very large. Using the Kernel trick to compute $\hat{K}$ reduces the computational effort from $O(N^2M)$ to $O(N^2P)$. This technique is known as the Kernel Trick. The decomposition of $\Psi(X)$ using SVD also fits in well with this idea because it is defined through the use of correlation matrices which are effectively combinations of inner products.

In the case of EDMD, a Kernel function f can be used to define basis functions for use in the Dictionary. There exist different functions for different classes of function bases. For example, there are polynomial and Gaussian Kernels [14]. A linear Kernel $f(x,y) = x^T y$ implicitly evaluates the space of linear functions as used in DMD [32]. When the computation of $\hat{A}$ and $\hat{G}$ is replaced with the kernel trick, $\hat{A}$ and $\hat{G}$ become kernel matrices defined by

$$G^{\hat{(ij)}} = f(x^{(j)}, x^{(i)}) \text{ and } A^{\hat{(ij)}} = f(y^{(j)}, x^{(i)}). \tag{2.43}$$

2.2.5 Eigenfunctions, Eigenvectors, and Modes

The next step is to find the eigenfunctions $P\Psi$ of K_P defined by

$$P\Phi = WP\Psi, \tag{2.44}$$

which is the same as equation 2.32 but with the eigenfunctions as a projection onto the space spanned by Q. The evaluated eigenfunctions of K_p can be calculated using only the elements of $\hat{K}$. This can be done with the left eigenvectors of $\hat{K}$ through eigendecomposition. Evaluating the eigenfunctions at X, the SVD of $\Psi(X)$ can again be used as well as the projection operator as $P = QQ^T$ to result in

$$P\Phi(X) = WP\Psi(X) = \hat{W}\Sigma Z^T = \hat{W}\Sigma^{-1} Z^T \hat{G}. \tag{2.45}$$

Through a similar process $\Psi(Y)$ can be replaced by

$$P\Phi(Y) = WP\Psi(Y) = \hat{W}\Sigma^{-1} Z^T \hat{A}. \tag{2.46}$$

The eigenfunctions evaluated at X and Y are now rewritten in terms of the matrices Σ and Z^T, $\hat{G}$, and $\hat{A}$, which can be calculated without any explicit evaluation of the dictionary functions using a kernel function and the relationship from equation 2.41. A similar manipulation can be done to find the relationship for the left eigenvectors V and $\hat{V}$. By definition, W is the inverse of V with right eigenvectors as columns (assuming full rank) (Kernel EDMD learning).

The Koopman modes ξ_k can be calculated similarly to the DMD modes in section 2.2.1. A sample point from the state space x can be recreated by evaluating all of the eigenfunctions at the data point, multiplying each eigenfunction by its respective Koopman modes for that data point, and taking the sum as

$$x = \sum_{k=1}^{K} \xi_k \phi_k(x). \tag{2.47}$$

Equation 2.47 can be rewritten for all sample points in X and Y and all eigenfunctions of K_P as

$$X = P\Phi(X)\Xi. \tag{2.48}$$

To calculate the Koopman modes for every eigenfunction, the pseudoinverse of $P\Psi(X)$ is taken as

$$\Xi = P\Phi(X)^{+}X. \tag{2.49}$$

Now, $P\Phi(X)$ can replaced with its equivalent from equation 2.45 which results in

$$\Xi = (\hat{W}\Sigma^{-1}Z^{T}G)^{+}X. \tag{2.50}$$

2.2.6 Kernel EDMD-DL

The best choice of dictionary elements and, by extension, the best choice of Kernel function to use in Kernel EDMD, is a topic of debate and is unique for each dynamical system [32]. Some kernels have free parameters which change the functions in their associated dictionary. For example, the RBF kernel contains a parameter θ, which influences the region of similarity between two radius centers. For some kernels, the free parameter values offer an opportunity to optimize the kernel to the data set and potentially reduce the complexity of finding the right choice of kernel function [19]. To take advantage of this optimization opportunity, a gradient learning method can be used to minimize the loss function in equation 2.40, as long as the loss function is differentiable in relationship to the free parameters.

To utilize the free kernel parameter learning method, the loss function must be rewritten to remove any explicit evaluations of the dictionary functions and replace them with the kernel function. This can be done by replacing the evaluations of the dictionary functions at X and Y with the relationship to the eigenfunctions from equation 2.44, using the property that V (a matrix with right eigenvectors of K_P as columns) is the inverse of W, and taking advantage of the idempotent property of P. Additionally, K_P is replaced by its eigendecomposition. This results in

$$J(K_P, f) = \|PVP\Phi(Y) - PV\Lambda(P\Phi(X))\|_F^2\,. \tag{2.51}$$

Next, using $Q^T V = PV = \hat{V}$, plugging in the relationship for the projected eigenfunctions to the left eigenvectors from equations 2.45 and 2.46, and removing Q because it is an isometry in Euclidean space when applied from the left, equation 3.7 becomes [24][3]

$$J(Kp, f) = \left\|\hat{V}\hat{W}\Sigma^{+}Z^{T}\hat{A} - \hat{V}\Lambda\hat{W}\Sigma Z^{T}\right\|_F^2\,. \tag{2.52}$$

Because the eigenvalues for K_P and $\hat{K}$ are the same and solving for K_P is not realistically possible, $J(Kp, f)$ is changed to $J(\hat{K}, f)$.

To learn a parameterized kernel, first $\hat{K}$ is calculated using a fixed dictionary, then holding the approximated $\hat{K}$ fixed, the parameters of the Kernel are updated and, by extension, the dictionary. Any gradient learning method can be used to update the kernel parameters. This method requires taking the derivative of the loss function with respect to the kernel parameters. This can be easy if $\hat{K}$ is constant during each update. There are many kernel functions with more than one parameter, which can all be learned using the same procedure. In addition to using a single kernel function, kernels can be added or multiplied.

As a final note, similar to what was done in [21], a regularization term is added to the loss function. The reason for this is to ensure while learning that the right-hand term in function 2.52 $\hat{V}\Lambda\hat{W}\Sigma Z^T$ never is equal to zero. If it does, the solution to the loss function could be trivial, and it is not guaranteed that the optimization problem finds a Koopman invariant subspace as described in section 2.1.3 [21]. The regularization term is the identity function of Y minus the predicted values by the Koopman approximation at Y as in

$$J(Kp, f) = \left\| \hat{V}\hat{W}\Sigma^{+}Z^T\hat{A} - \hat{V}\Lambda\hat{W}\Sigma Z^T||_F^2 + ||Y - \Xi\hat{W}\Sigma^{-1}Z^T\hat{A} \right\|^2. \tag{2.53}$$

In equation 2.53, the Koopman modes are found for the values in Y (not for the values in X as described in section 2.2.5). The Koopman modes are then multiplied by the eigenfunctions evaluated at Y. If the trivial solution is found with the left part of the loss function then the newly added term will penalize the loss.

2.2.7 Recreating the System Dynamics

A summary of the complete algorithm is described in algorithm 1.

Algorithm 1 Kernel EDMD-DL Algorithm

```
Initialize $K, \theta$
Set learning rate $n > 0$
Set the maximum number of Epochs $E > 0$
while Epochs $< E$ do
    $\hat{A^{(ij)}} \leftarrow f(y_j, x_i)$
    $\hat{G^{(ij)}} \leftarrow f(x_j, x_i)$
    Compute $\hat{G} = \Sigma^2 Z^T$
    Compute $\hat{K} \leftarrow (Z\Sigma^{+})^T\hat{A}(Z\Sigma^{+})$
    Compute eigenvalues/vectors $\hat{V}, \hat{W}, \Lambda$
    $\theta \leftarrow \theta - n\nabla_\theta||\hat{V}\hat{W}\Sigma^{+}Z^T\hat{A} - \hat{V}\Lambda\hat{W}\Sigma Z^T||_F^2$
end while
```

In algorithm 1, a while loop cycles over a specified number of training epochs, or until the loss is below a threshold. Within each iteration, $\hat{A}$ and $\hat{G}$ must be calculated using the kernel according to 2.2.4. Then, eigendecomposition is performed on $\hat{G}$ and $\hat{K}$ is calculated from $\hat{A}$ and the eigendecomposition of $\hat{G}$ as discussed in section 2.2.3. Next, the eigenvectors and eigenvalues of $\hat{K}$ are found and used as part of the loss function $J(\hat{K}, \theta)$ as described in sections 2.2.5 and 2.2.6. Holding the components of $\hat{K}$ constant, a gradient learning method can then be used with this loss function to update the kernel parameter θ as discussed in section 2.2.6.

Once a suitable $\hat{K}$ has been calculated, using Algorithm 1, it can be used to predict the original underlying dynamic system. This can be done through the spectral decomposition of $\hat{K}$ by using equation 2.11. First, an initial state vector x_0 is chosen . This could either be a snapshot from the data or it could be a new state vector. The accuracy of the latter, however, greatly depends on the relationship of the new initial state vector to those the $\hat{K}$ was constructed on. This topic will be discussed in more detail in section 3.1.2.

Once an initial state vector is chosen, it is used to calculate the Kernel matrix $\hat{G_{x_0}}$ between X and x_0 as

$$\hat{G_{x_0}}^{(ij)} = f(x_0^{(j)}, x^{(i)}). \tag{2.54}$$

Next, the eigenfunctions of $\hat{K}$ are evaluated at the new state vector x_0 using

$$P\Phi(x_0) = \hat{W}\Sigma^{-1}Z^T\hat{G_{x_0}}. \tag{2.55}$$

The next step is to solve 2.50 to uncover the Koopman modes by using the kernel matrix $\hat{G}$ calculated between X and X, not G_{x_0}. Finally, as in equation 2.11, calculate the predicted value of x_1 at the next timestep by multiplying together the Koopman modes, the eigenvalues of $\hat{K}$, and $P\Phi(x_0)$ as

$$x_{n+1} = \Xi\Lambda P\Phi(x_0). \tag{2.56}$$

To advance the system more timesteps, repeat the same procedure using the newly calculated state vector $x(n+1)$ as the initial state vector x_0 until the desired end of the trajectory has been reached. As with the choice of the initial state vector, the duration of the predicted dynamics, that can still be accurately calculated, depends highly on the relationship of the prediction to the data used to construct $\hat{K}$ and the underlying dynamical system. It should be noted that $\hat{K}$ can also advance the linear system. This method is generally more robust, but using the spectral decomposition method is computationally cheaper. In this thesis, the spectral decomposition method is used.

3 Applications of Kernel-Based EDMD

3.1 Experimental Methodology

This section discusses the methodology used to apply the kernel EDMD to two ODE systems of differing complexity. The first system is an example of a Hopf bifurcation, and the second is a more complex system representing hydrogen combustion in a Continuous stirred tank reactor (CSTR). What is of interest is recreating the dynamic systems over a wide range of initial starting conditions and prediction of trajectories between sampled data points. This scenario mimics a real-world application where experimental data is collected from a physical dynamic system, and the goal is to use this data to simulate the system's behavior with conditions that were not directly tested during experimentation to enhance understanding of the system efficiently. Python is used as the base coding language to implement EDMD, and the PyTorch library is used specifically for kernel learning. In the following sections, the experimental design is described in detail, including the data collection, accuracy assessment, kernel selection, and hyperparameter tuning.

3.1.1 Data Collection

The data for both dynamical systems is generated numerically. Discrete solutions to the dynamical systems are found using numerical solvers. For each system, separate trajectories are calculated at several initial conditions distributed over the solution space of interest. A trajectory is defined here as the path of a system state over time. An example of this can be seen in figure 3.1, which depicts 4 separate trajectories of the cubic Lotka-Volterra equations over a time span of t=0 to t=0.4 (unitless). Each trajectory is a separate color and the initial condition is labeled with a corresponding colored dot. The direction of movement of the path over time is labeled with an arrow. The system states in this example are y_0 and y_1, and their evaluations in time are defined by the equation system 3.6. The leftmost graph is of y_0 plotted vs y_1 and the other graphs are of each parameter plotted against time.

The discrete-time interval is set small enough initially for the solver to solve the numerical system accurately. A subset of that data distributed evenly among the entire trajectory is then used for input into the kernel EDMD methods. When using machine learning techniques in general, it is important to have quality input data. Certain characteristics in the data are necessary to calculate accurate predictions using the kernel EDMD-DL method. First, the time steps must have sufficient granularity to capture the desired detail in the system. Additionally, the magnitude of changes in the dynamic system in each dimension

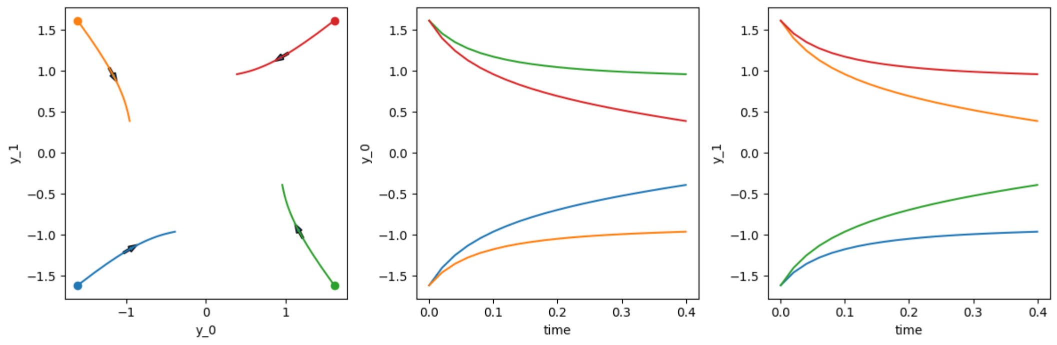


Figure 3.1: Example of 4 trajectories with separate initial conditions plotted over $t = 0$ to $t = 4$ for the Hopf bifurcation equation system.

should be relatively similar to ensure the numerical system results in a well-conditioned numerical method in all dimensions. One way to ensure this is the case is to normalize the data or apply transformations. Finally, it should be avoided that the observed trajectories should not overlap each other. This is because, when this situation occurs, there is more than one potential "dynamic path" for a state vector to evolve to in the next time step. The reason for this is that the dimensionality of the training data is not high enough to explain the dynamic behavior.

Sometimes, it is impossible to avoid regions of overlap or to increase the dimensionality of the sample data. An example of this is in a real-world chemical experiment, where the spectroscopy equipment is unable to adequately measure the concentrations of all intermediate species, which drives the rate of reaction. In this case, time delay methods could be used as a solution; however, this method is not discussed in detail in this thesis.

3.1.2 Reporting Accuracy

The loss function defined in equation 2.52 is used to quantify the error of the reconstructed in-sample trajectories from the data set. Additionally, three out-of-sample (OOS) trajectories are predicted with the same ending and starting time as the in-sample trajectories on which the model is trained. An in-sample trajectory is a trajectory with the same initial condition as a trajectory from the original data set that is used to calculate the approximate Koopman operator through the kernel EDMD-DL process. An OOS trajectory is defined as a trajectory with an initial condition not contained within the original data set. To ensure the accuracy of an OOS trajectory, the initial conditions of the OOS trajectory are placed between the initial conditions used for the in-sample trajectories. This is because the dynamic system might behave much differently outside the bounds of the data set used for training, and the approximated Koopman operator only estimates how the system behaves in between the data samples used for training.

The accuracy of an OOS trajectory is calculated using the averaged L2 norm over all time steps of the true trajectory values of the dynamic system minus the predicted trajectory values. A new true trajectory with the same initial condition is also calculated from the original system to compare. The OOS trajectory error is calculated with

$$\text{OOS error} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} |x(n) - \tilde{x}(n)|^2}, \tag{3.1}$$

where N is the number of time steps, $x(n)$ is the true value at time step n, and $\tilde{x}(n)$ is the predicted value at time step n. Using the OOS trajectories to measure the error is comparable to a simple form of cross-validation. For each example, 3 OOS trajectories are be calculated with unique randomly distributed initial conditions. The final OOS error reported is the error of all 3 trajectories. This is still referred to as just the OOS error.

An acceptable in-sample and OOS error is highly dependent upon the dynamical system and purpose of prediction. Of course, the sample data can include experimental noise or numerical errors when compared to the true behavior of the dynamic system it sampled from. Here, the focus is only on the capabilities of the Kernel EDMD-DL method assessed with comparison to the sample data. The values of all data used are between 0 and 2.5, so any reported error values below $1e^{-15}$ are considered to have machine precision.

3.1.3 Kernel Selection

As covered in section 2.2.6, the best choice of kernel function that achieves optimal Kernel EDMD results is highly dependent upon the system. Finding the correct choice of basis functions is difficult. If the mathematical equations governing the underlying system are known, this might be a simpler task, but this is not the case in most cases.

To choose the best dictionary of basis functions for the systems presented here, a ”tried and true” approach is taken. The application of the linear and cubic polynomial kernels to the Hopf system bifurcation has already been documented [20] in literature and the use of these kernels serves as a great proof of concept to demonstrate the capabilities of the kernel EDMD method. Additionally, no kernel parameter learning is necessary. The linear kernel matrix is [2]

$$K(X, Y) = X^T Y \ . \tag{3.2}$$

The cubic polynomial kernel matrix is [14]

$$K(X, Y) = (X^T Y + 1)^3 \ . \tag{3.3}$$

Technically, the order of the cubic polynomial can be changed to any desired order a, and this is a learnable hyperparameter of the kernel, but in this case, it will be kept constant at a value of 3 and the kernel will be treated as if it is parameter-free. The cubic kernel contains in its implicit dictionary all polynomials up to and including a degree 3 polynomial [32].

To perform kernel EDMD-ML, on both the Hopf bifurcation and the CSTR combustion reaction examples, the Standard Radial Basis Function kernel(RBF) (also known as the Gaussian kernel) is used. This kernel is chosen because it is known to provide good results for a wide variety of systems [32][21][5][31]. The possibility exists that a different kernel could yield more accurate results for the specific systems under consideration; nevertheless, this idea remains unexplored.

The RBF kernel is [2]

$$K(X,Y) = \exp\left(\frac{-\parallel X - Y \parallel^2}{2\theta^2}\right) . \tag{3.4}$$

The free parameter θ is called the bandwidth and is a trainable parameter. The RBF kernel calculates the distance between two data points. The farther apart the points are, the smaller the value of the kernel. The RBF kernel is a stationary homogeneous kernel, meaning the kernel's value is relative only to the distance between points in the two input matrices [2].

When using the RBF kernel in Kernel EDMD-DL, the radius centers are located at the sample points, so the more sample points that are used, the greater the number of radius centers. The trainable free parameter θ affects how quickly the function decreases as the distance between points decreases. A similar phenomenon happens with the polynomial kernel because each data point adds more terms to the polynomial expansion. Increasing the number of data points increases the number of basis functions in the dictionary, which expands the predictive capabilities of the EDMD dictionary, but can also lead to overfitting of the data due to an increase in the condition numbers of $\hat{G}$ and $\hat{A}$ matrices. If too high, the calculated Koopman operator approximation can amplify errors and poorly fit OOS trajectories.

3.1.4 Hyperparamter Optimization

There are many different hyperparameters that influence the accuracy of the OOS and in-sample accuracy measures. Taking a strategic approach to hyperparameter optimization is of utmost importance, as it can determine the feasibility and effectiveness of a method. The most important factors for both the OOS and in-sample accuracy are:

- Number of time steps in each trajectory
- Number of eigenfunctions associated with the approximated koopman operator
- Value of the kernel free parameter

The last item is only applicable to Kernel EDMD-ML. It should be noted that the number of time steps is not always a changeable hyperparameter. For example, if the sample data is experimental, the number of data points collected is often limited by the equipment used for the experiment. In this case, the number of time steps is fixed. Here, however, it is easy to do with the numerically generated data so we consider time steps to be an adjustable hyperparameter to gain better insight into how the value affects the accuracy of the results. This knowledge could be used to make informed decisions about whether or not the kernel EDMD-DL method is appropriate for certain situations.

The sensitivity that the results have to each factor depends highly on the complexity of the underlying system. Additionally, when predicting the dynamics of the OOS trajectory, the accuracy of the predicted system is influenced by the number and the range of the initial conditions. The number of initial conditions which can be tested is limited by computation time, but the trajectories must be close enough together for the Koopman operator to have enough information to capture the dynamics between the known information. If the trajectories are too close, this can also cause problems because it increases the sensitivity of the prediction to the other hyperparameters. The number of initial conditions and the state space range of the trajectories also affect the optimal values for the number of time steps, the number of eigenfunctions, and the kernel free parameters.

The hyperparameter optimization strategy is to first define the state space of interest and choose initial conditions that produce trajectories that fully cover the space. The trajectories should ideally all be part of the same bifurcation, and the more initial conditions, the better. This, of course, is dependent upon the allowed computation time.

Next, once the dynamic system data has been generated, a grid search is performed for a large hyperparameter range. The grid search includes the number of time steps and the number of eigenfunctions for the linear and polynomial kernels, which do not have any free kernel parameters. For the RBF kernel, with a trainable kernel free parameter, the number of time steps is set constant before the grid search. The search is then performed using different combinations of the number of eigenfunctions and the free parameter.

The reason for using both the gradient learning method of Kernel EDMD-ML and the grid search to optimize the kernel free parameter is because, based on experience, the gradient method for the free parameter training is very sensitive to the initial value set at the beginning of training. It is, therefore, crucial to have at least a basic understanding of how the free parameter affects the accuracy of the results to choose an appropriate initial condition.

During the grid search, for each parameter combination, kernel EDMD-DL is performed on the same sample data, and the OOS error and in-sample loss are calculated. The optimal results are found by locating the parameter combination that effectively minimizes both accuracy measures. The metric employed to consolidate these two accuracy assessments into a single value, facilitating comparisons across the entire grid search, is the simple Root Mean Square Error (RMSE) defined as

$$\text{combined error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}. \tag{3.5}$$

Once the optimal parameter settings are detected, a simple guess-and-check methodology is used to fine-tune the values with higher granularity. Admittedly, this procedure is not technical or reproducible, but in most cases is faster than performing another grid search with a smaller parameter range. The overall grid search strategy taken here is rather rudimentary. For this reason, the optimality of the hyperparameters cannot be guaranteed when any results are reported.

Unlike the number of eigenfunctions and the number of time steps, kernel-free parameters can be learned using a gradient method combined with the in-sample loss function as described in section 2.2.6. The gradient learning procedure introduces a whole new set of additional hyperparameters in addition to the initial value for the free parameter already discussed. These include the learning rate, the optimizer, and the number of training epochs. Also, depending upon the optimizer chosen, even more parameters can be introduced, such as the weight decay if RMSprop is used. The strategy used here is to first run the training with basic default settings and the classic gradient descent method. The learning rate is set to $1e^{-1}$ and the number of epochs is set to 100. The training landscape is then analyzed and adjustments to the parameters are made on a system-specific basis.

The gradient learning process only utilizes the in-sample loss update of the free parameter and does not take into account the OOS error. It makes sense that this learning process should also include some cross-validation capabilities. The search for the optimal kernel-free parameter could be made more efficient if the calculation of the trajectory error can also be incorporated into the learning somehow. The standard form of cross-validation, where the data is broken up into randomized test and validation groups, does not work in this case because it is important to keep the temporal and spatial structure of the X and Y matrices intact. However, it is feasible to split the data into groups of trajectories. Unfortunately, this is not incorporated into the gradient learning method, so the results reported for the parameter learning process are more a proof of concept rather than a full optimization technique. To find a true optimal kernel-free parameter value, it is also required to compare the result of the learning with the results from the grid search.

3.1.5 Eigenfunctions and Eigenvalues

Probably the most critical parameter which influences the accuracy of the kernel EDMD results is the number of eigenfunctions that are retained during the computation of the approximated Koopman operator. Theoretically, the more eigenfunctions with non-singular eigenvalues that are retained, the more accurate the results; however, many eigenfunctions have small singular values which are not significant or unstable due to numerical errors. This means there is an optimum number of eigenfunctions with leading singular values that must be retained to achieve the highest prediction accuracy. On either side of the optimum, the prediction error increases. The more complex the underlying dynamical system, the more sensitive the results become to the number of eigenfunctions. The proper number of eigenfunctions to retain is dependent on the number of sample points from the underlying system used to construct the Koopman approximation (trajectory time steps and initial conditions) and the type of functions the dictionary comprises.

Many strategies exist to find the proper number of eigenfunctions. One strategy is to apply the kernel EDMD-DL procedure to subsets of the data, compare the resulting eigenvalues, and remove ones that are not consistent [31]. Another strategy is the evaluate the quantity of the eigenfunctions by comparing the error associated with each eigenfunction when it is individually applied to the state space and using Monte-Carlo integration to form an error estimate [21]. The method used here is much more rudimentary. Rather, a holistic view is taken and the appropriate number of eigenfunctions is found by searching for the minimum of the OOS error and the in-sample loss. Ideally, this would also optimize the combined error defined in equation 3.5.

3.2 Application of Kernel EDMD to Hopf Bifurcation

3.2.1 Hopf System Background Theory and Data Collection

This first dynamic system is comprised of the cubic Lotka-Volterra equations, which are simple two-dimensional coupled ODEs that exhibit an example of a Hopf bifurcation. The equations describe the rate of change of two populations over time with a predator-prey relationship defined as

$$\begin{aligned} \frac{dy_0}{dt} &= -y_1 + y_0(\mu - y_0^2 - y_1^2) \\ \frac{dy_1}{dt} &= -y_0 + y_1(\mu - y_0^2 - y_1^2). \end{aligned} \tag{3.6}$$

In this system, the parameter μ determines whether the system is a fixed point or a limit cycle. When μ is set to 1, the solution to the ODE system is a limit cycle. This can be seen in figure 3.2 under the ”Original Numerical Solution” column.

The numerical solution to the ODE system is found using Scipy’s ODE solver, which uses the adaptive explicit Runge-Kutta method of order 5(4). The error is controlled with a 4th-order accurate method, but the steps are taken with a 5th-order accurate formula. The solver calculates an estimate for the local error, and the time step is either decreased to reduce the local error or increased to speed up the calculations if the error is under a threshold [12]. The default values of $1e^{-3}$ and $1e^{-6}$ for the relative and absolute local error tolerances are used. When the time interval of integration is passed into the function, as is done in this case, the function interpolates the variable time step solution to the desired input time steps allowing control of the number of data samples collected from the system.

The Runge-Kutta 5(4) solver is good for non-stiff initial value problems and is relatively fast. To highlight the capabilities of the Kernel EDMD methods, the accuracy of the numerical solution is not of critical importance. The essential aspect is that the solver is capable of generating the anticipated bifurcations and providing a solution at designated time intervals.

The ODE system is completely defined by the states (y_0 and y_1) and it is assumed both states can be set to any value in the state space as an initial condition. Because of this, many short trajectories over many initial conditions can be generated. This enables an efficient method to gain information about the underlying dynamic system evenly over the state space of interest. Additionally, because the Hopf bifurcation is symmetrical in both dimensions around $(0, 0)$, no data normalization is required. Finally, the dynamics do not display multiscale time dynamics across either dimension along the trajectory path. This means a transformation of the data is not required.

The linear, polynomial, and RBF kernels are applied to model the system dynamics. The basic Kernel EDMD method is used with the linear and polynomial Kernels, and the Kernel EDMD-ML method is used with the RBF kernel. Within a $(-2, 2)$ interval for y_0 and y_1, 64 initial conditions are distributed evenly across the parameter space, with trajectories running from t=0 to t=0.4.

3.2.2 Hopf System Results

Figure 3.2 shows the combined results for the linear, polynomial, and RBF kernels together. While no claims are made about complete optimality, the figure shows the results from the best hyperparameter combinations found for each kernel. The top left graph in figure 3.2 shows the 64 trajectories from the original numerical solution that the data samples collected.

The second row of figure 3.2 shows the predictions of three OOS trajectories with randomly chosen initial values at $(1.6, 1.6), (0.75, 0.75), (-0.4, -0.4)$ and a trajectory end time of 2.0 seconds. The trajectory end time is extended only for visualization purposes, and only values up to 0.4 seconds are used for the accuracy calculation. This allows for a better comparison of the OOS to the loss from the training data.

Figure 3.2 shows that the RBF kernel resulted in the best predictions, and the results from the polynomial kernel are a close second. The linear kernel could not recreate the limit cycle bifurcation and instead predicted a stable point. The cause of this is the inability of the set of linear basis functions defined directly on the states to recreate the complex dynamics of the Hopf bifurcation. The stable point is the closest approximation to the underlying system that the linear basis functions could form. Visually, the difference between the linear kernel results and the polynomial and RBF kernel results is easy to see. The accuracy values reported here are comparable but slightly higher than those reported in the literature for similar well-studied nonlinear functions [21][1]; however, it can be difficult to directly compare accuracy reports due to the wide variety of accuracy measures.

An important aspect to discuss is the number of eigenfunctions required for the reconstruction, which relates to the computational power needed. The polynomial kernel requires only 10 eigenfunctions, while the RBF kernel requires 25 to achieve nearly the same accuracy for both the in-sample and OOS trajectories. One might expect the opposite because the shape of the radial basis functions is the same as that of the Hopf bifurcation. The radial basis functions could be used to linearize the phase space's manifold [20]. This can only be the case when the centers are placed intentionally in the center of the bifurcation. When using this method of Kernel EDMD-DL and the RBF function with the trainable θ parameter, the radial centers must be placed at points along the trajectories. In the case of

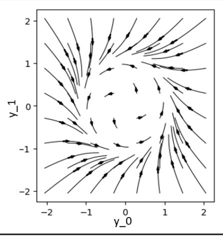
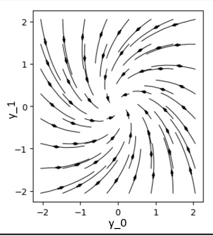
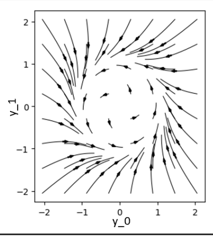
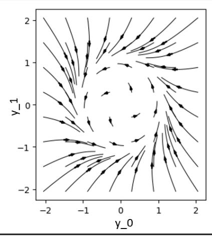
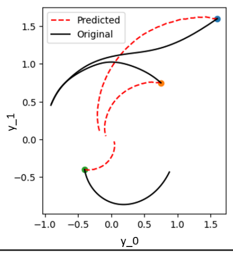
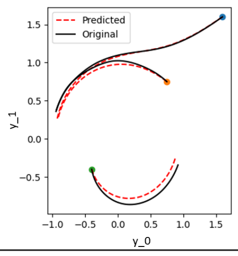
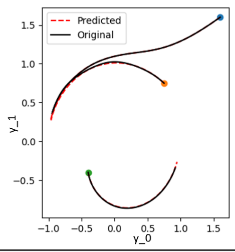
Original Numerical Solution	Linear Kernel	Polynomial Kernel	RBF Kernel
			
Out-of-Sample Trajectories Initial conditions [1.6,1.6], [0.75,0.75], [-0.4,-0.4]			
Hyperparameters	Time Start = 0 sec Time End = 0.4 sec Number of Timesteps = 20 Number Eigenfunctions = 5	Time Start = 0 sec Time End = 0.4 sec Number of Timesteps = 20 Number Eigenfunctions = 10	Time Start = 0 sec Time End = 0.4 sec Number of Timesteps = 20 Number Eigenfunctions = 25 Free parameter Θ =12.074
Accuracy	In-Sample Loss = 0.0161 Average OOS Error = 0.594	In-Sample Loss = 5.296e-09 Average OOS Error = 0.0704	In-Sample Loss 2.012e-11 Average OOS Error = 0.057

Figure 3.2: The original numerical solution for the Hopf bifurcation system compared to predicted kernel EDMD results for the linear, polynomial, and RBF kernels.

the Hopf bifurcation, this greatly complicates the reconstruction and requires more eigenfunctions.

3.2.3 Hopf System Hyperparameter Optimization

The linear and polynomial kernels do not have any parameters which can be trained through a gradient method like the RBF kernel. Therefore, a grid search is performed to better understand how the results' accuracy changes with the number of time steps for each trajectory and the number of leading eigenfunctions. The bounds of the grid search are chosen differently for each system, and only an interesting subset of the results are presented. Figure 3.3 and 3.4 show the grid search results for both the OOS error and the in-sample loss separately for the linear kernel and polynomial kernel, respectively.

The data from the grid searches for both kernels shows that the OOS trajectory loss is lower for most parameter combinations but that both accuracy calculations show similar patterns. Both kernels show a steep decrease in the error after a cut-off number of eigenfunctions but an increase again after a certain number of eigenfunctions. The trend is even more prominent for the average OOS trajectory loss, which is a typical indication of over-

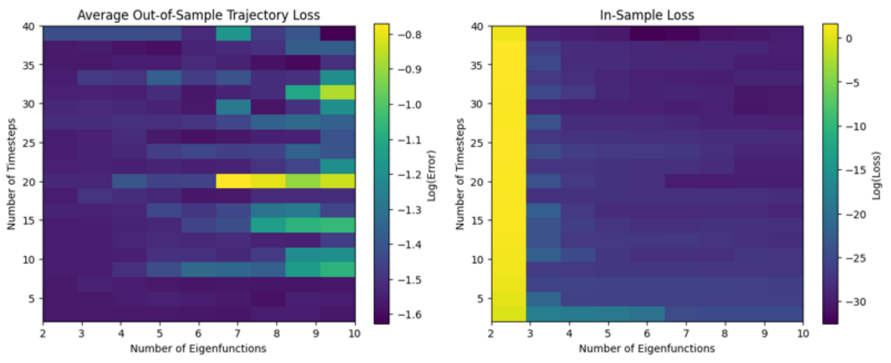


Figure 3.3: In-sample loss and OOS trajectory error for different combinations of Time steps and Eigenfunctions using the linear kernel with the Hopf bifurcation example.

fitting. This analysis confirms there exists an optimum point in the parameter space where the loss is minimal. This optimal point, however, can be challenging to find and anticipate from the system if an initial grid search is not performed.

Based on the grid search results, the combination error from equation 3.5 for the linear kernel located the best parameters to be 4 time steps and 5 eigenfunctions, and for the polynomial kernel 20 time steps and 10 eigenvalues.To better compare the results, 20 time steps are chosen for all kernels despite the results of the combined error. For the linear kernel, the two different timesteps are tested and there was no significant change in the OOS trajectory error or the in-sample loss by increasing the number of time steps and keeping the number of eigenfunctions constant.

3.2.4 Learning the Parameterized RBF Kernel Using Kernel EDMD-DL

A grid search is performed, setting the number of time steps to 20, and varying the kernel free parameter θ and the number of eigenfunctions for the RBF kernel. This produces a minimum combined error at a θ value of 6.0 and 32 eigenfunctions. Another nearly comparably low combined error parameter combination was 10.0 and 25 eigenfunctions. This parameter set was chosen instead for the next steps the smaller number of eigenfunctions was slightly faster computationally. Figure 3.5 shows the grid search results. Based on the results, the number of eigenfunctions is set constant at 25. To gain more detailed insight into the loss landscape for θ when the number of eigenfunctions is held constant, figure 3.6 is created. This time only the in-sample loss is calculated for θ values between 1 and 50.

Analyzing figure 3.6, an obvious global minimum can be seen at a θ value of around 11. Anywhere between a θ value of 5 and 12 is nearly at machine precision and would be considered an acceptable result. What is surprising, however, is the roughness of the

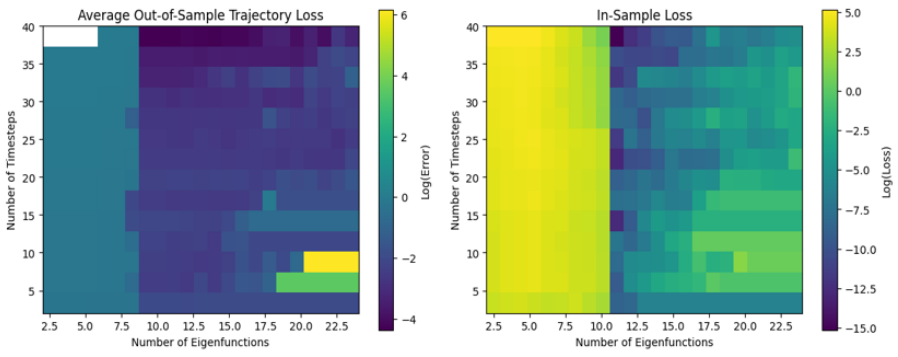


Figure 3.4: In-sample loss and average OOS trajectory error for different combinations of Time steps and Eigenfunctions using the polynomial kernel with the Hopf bifurcation example.

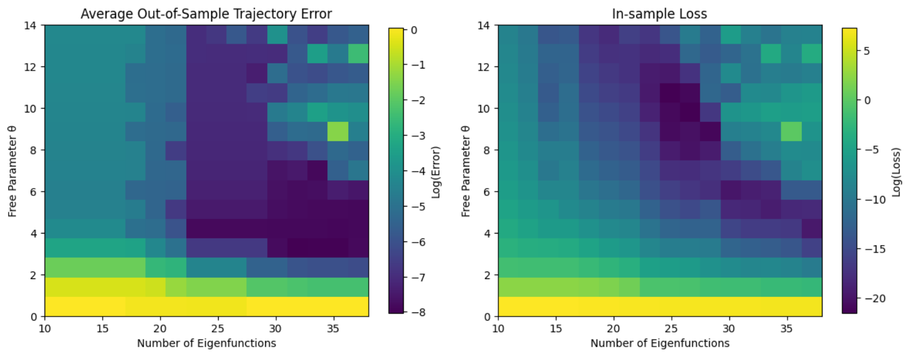


Figure 3.5: In-sample loss and OOS trajectory error using RBF Kernel for different combinations of the free parameter θ and eigenfunctions for Hopf bifurcation example.

landscape and the many local minima. This indicates that the gradient learning algorithm will probably have difficulty converging to the global minimum depending on the location of the starting θ value.

Learning of the θ parameter is attempted using a classic batch gradient descent optimizer with a learning rate of $1e^{-1}$, 100 epochs, and an initial value of 2.0. Figure 3.7 visualizes the in-sample loss vs. epoch as a result of the training. After 100 epochs, the final θ value is calculated to be 2.592, corresponding to a loss value of $9.359e^{-4}$ after 20 minutes. The in-sample loss result is very acceptable. However, obviously based on 3.6, the global minimum has not been reached. The results of this experiment show that the gradient method does have the potential to find the optimum; however, more optimized learning parameters and a different optimizer are needed to converge to a better parameter value in such a rugged loss landscape in an efficient amount of time.

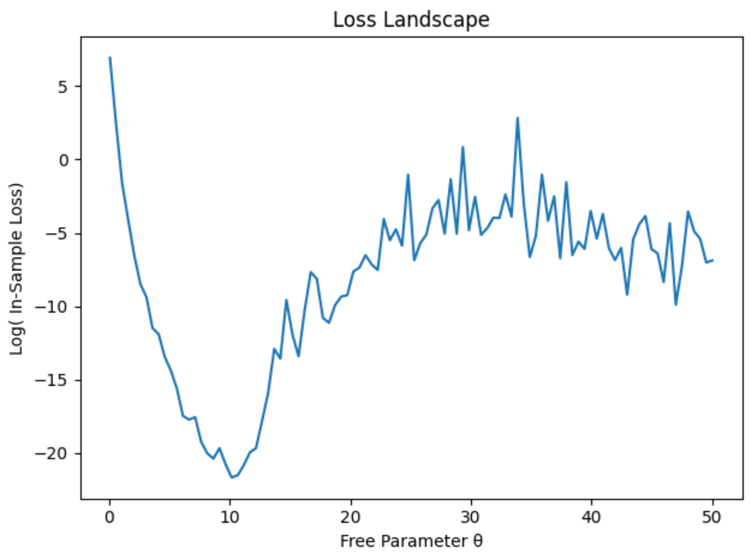


Figure 3.6: In-Sample Loss vs. Free Parameter θ for Hopf bifurcation example with 25 eigenfunctions and 20 time steps.

As a second attempt, RMSprop optimizer learning optimizer is used with a learning rate of $1e^{-1}$, momentum=0.9, 50 epochs, default weight decay of 0, and an initial value of 2.0. RMSprop stands for root mean square optimization and is an adaptive learning rate method that is a more robust optimizer designed to produce better results in a rough landscape with a local minimum. It uses a moving average of the squared gradient to scale the learning rate, which helps stabilize the learning process in a rough loss landscape.

Figure 3.8 shows the results for the free parameter learning with the RMSProp optimizer. Compared to 3.7, the learning is much smoother, and convergence to the optimum θ value is achieved in far fewer iterations. Only the first ten epochs are displayed in the figure because any changes after that are visually too difficult to discern. The final θ value after 50 epochs is calculated to be 12.074. This value results in a loss of $2.012e^{-11}$, which is a very good improvement from the loss at a θ value of $9.359e^{-4}$. No further by-hand adjustments are made afterward to the θ because changing the number of eigenfunctions and θ by a small amount in either direction does not significantly improve the accuracy.

3.3 Application of Kernel EDMD to Hydrogen Combustion CSTR

For the most important part of this thesis, the capabilities of kernel EDMD-DL are showcased in a practical application to the dynamics of hydrogen combustion in an ideal gas, constant volume, continuous stirred tank reactor (CSTR) with perfect mixing. The overall goal is to use kernel EDMD-DL to accurately predict the system's behavior with initial

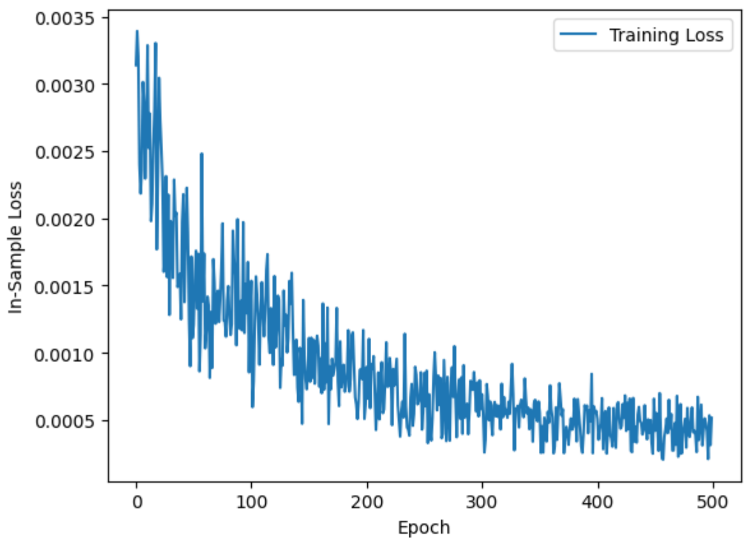


Figure 3.7: In-Sample Loss vs. Free Parameter θ for Hopf bifurcation data with 25 eigenfunctions and 20 time steps using gradient descent optimizer.

conditions outside of the ones used to generate the sample data. This particular dynamical system is chosen because the rate of chemical reactions in a CSTR reactor system can be modeled simply through a system of ODEs. If the CSTR is designed to maintain volume and the gasses are assumed to be ideal, then important assumptions can be made, significantly simplifying the reaction model. Hydrogen as a future fuel is currently an important topic. Although the reaction system used in this thesis is quite simple, it highlights the potential for the kernel EDMD methods to be extended to more complex real-world systems. The capability to numerically predict and simulate the dynamics of a hydrogen combustion system in a data-driven way can potentially accelerate future developments. Some examples of this are better control, decreasing experimental costs, increasing safety, or expanding the knowledge of where and how hydrogen combustion can be useful.

3.3.1 Hydrogen Combustion Background Theory

Hydrogen is considered to be a prime candidate for future fuel sources. Hydrogen is a clean fuel that stoichiometrically produces only water when burned. It can also be produced with renewable energy sources such as through electrolysis [29]. Hydrogen is an easily transportable fuel that can be used for many different mobility applications. Currently, the most promising application is the use of hydrogen in fuel cells which can generate energy for electric vehicles. Another application is direct fuel for internal combustion engines. Direct hydrogen combustion is less efficient and when burned at high temperatures in air can produce nitrogen pollutants in the form of nitrous oxides (NOx). [29]. For

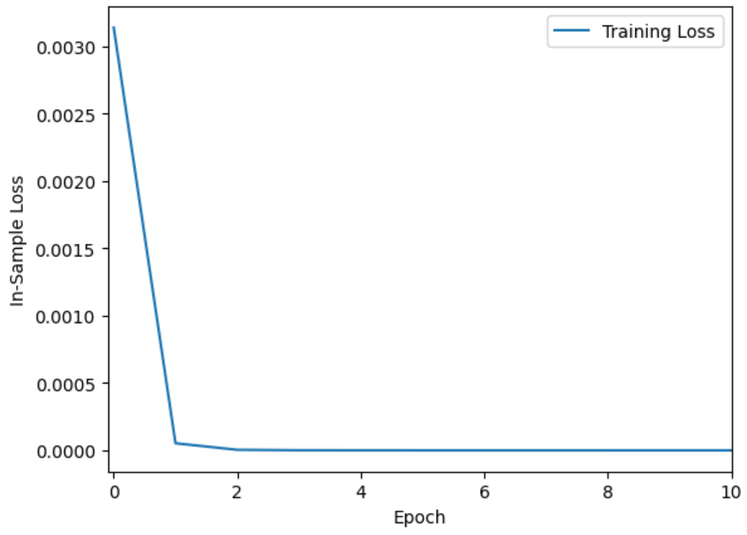


Figure 3.8: In-Sample Loss vs. Epoch for Hopf bifurcation example data from with 25 eigenfunctions and 20 time steps using RMSprop optimizer.

use with vehicles, it can be stored as a compressed gas and other ways of safer storage are currently being researched. One of the biggest problems with hydrogen combustion is that neither compressed nor liquid hydrogen can meet the current energy density storage capabilities of regular gasoline [29].

Hydrogen releases more energy, has a higher automatic ignition temperature, and more rapid combustion than gasoline. Combustion can also be performed with a lower air/fuel ratio. Compared to gasoline, hydrogen has a broader range of flammable concentrations. The lower flammability concentration generally increases with temperature and pressure. The auto-ignition temperature (the minimum temperature required to ignite a gas or vapor in air without a spark or flame being present) at standard atmospheric pressure (1 atm or 101.325 kPa) is disputed to be around 773 to 858 K [29].

When hydrogen is burned, the hydrogen directly reacts with oxygen

$$2H_2(g) + O_2 \rightarrow 2H_2O(g) + \text{energy}. \tag{3.7}$$

. 3.7 is the total stoichiometry of the combustion but when combustion occurs the actual reaction is much more complicated because the reaction includes third bodies, reaction intermediates, and side reactions. This is referred to as the reaction system. Reaction intermediates are species that are formed and consumed throughout the combustion. When combustion occurs in the presence of air or other gases other than oxygen, some gases can act as third bodies which are not consumed in the reaction but can affect the reaction rate. There has been much research on this topic to define correctly all of the intermediate reactions [11][23].

The reaction system used to generate numerical data is a common hydrogen-oxygen sub-mechanism from GRI-Mech 3.0 created by the Berkeley combustion team, which includes 11 species and 23 unique reversible elementary reactions [11] [23]. The 11 species are:

$$H_2 \quad H \quad O \quad O_2 \quad OH \quad H_2O \quad HO_2 \quad H_2O_2 \quad AR \quad N_2$$
.

The entire GRI-Mech 3.0 system is optimized for natural gas combustion and contains 53 species with 325 reactions. Here a subset of those reactions for hydrogen combustion is used.

3.3.2 CSTR Reactor Background Theory

CSTRs are used for continuous chemical production but can also be considered a significantly simplified version of an internal combustion engine. CSTR reactors have a constant volume with an inlet and an outlet that can be set to specified flow rates. The inlet contains the reacting species, and the outlet contains a mixture of reacting species and the products. The volume of the CSTR is fixed, and the time a packet of fluid spends in the reactor is called the residence time and it is a function of the reactor volume and the inlet flow rate. The longer the residence time, the more time the reaction has to proceed inside the reactor, and the higher the concentration of products in the outlet stream. Figure 3.9 is a descriptive illustration of a CSTR reactor.

When the reaction occurs inside the reactor, mass and energy are conserved so critical state variables can be calculated using complex mass and energy balances throughout the reaction. In the simplified CSTR model, at each moment, the species concentrations and the temperature drive the speed of the reaction. During combustion, the amount of each species, the pressure, and the temperature of the reactor change. CSTRs can also be insulated or fitted with heat exchangers to regulate the amount of energy that leaves the reactor as heat. For combustion reactions, removing heat energy is very important for safety reasons and the ability to control the extent of the reaction. Because autoignition only occurs at high temperatures, the entire reactor is placed into an oven which sets the inlet, outlet, and surrounding air temperatures to a specified value.

3.3.3 Defining and Solving the Dynamic Reaction System

The dynamic system describing the hydrogen combustion reaction for $i = 1...n_s$ species and $k = 1...n_r$ reactions in a constant volume CSTR is [10][18]

$$V\frac{d[X_i]}{dt} = Q_{in}[X_i^0] - Q_{out}[X_i] - V\dot{w}_i \tag{3.8}$$

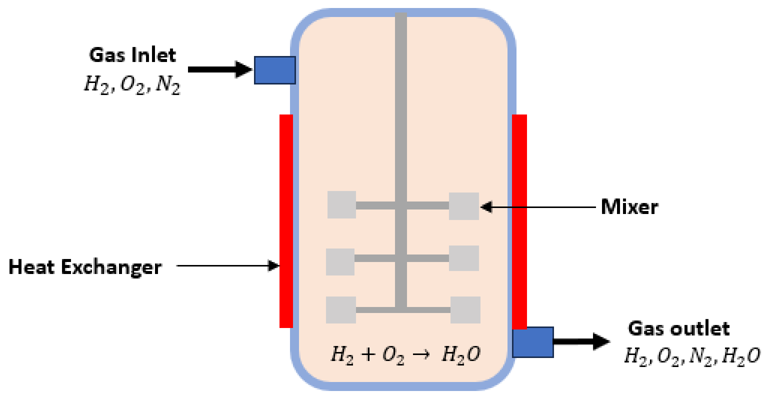


Figure 3.9

$$V \sum_{i=1}^{ns}([X_i]C_{v_i})\frac{dT_r}{dt} = Q_{in}\sum_{i=1}^{n_s}[(H_i^0 - H_i)[X_i^0]] - V\sum_{k=1}^{nr}(\Delta_r H_k q_k) - UA(T_r - T) + RT_r V\frac{d[X_{tot}]}{dt}, \tag{3.9}$$

where Q_{in} and Q_{out} are the volumetric flow rates of the inlet and outlet, respectively, T_r is the reactor temperature, and T is the surrounding oven temperature. ^0_i is the molar concentration, and H_i^0 is the total enthalpy of species i in the inflow. For the contents inside the reactor and in the outlet for each species i, C_i is the molar concentration, $\dot{w}_i$ is the production rate, C_{v_i} is the molar heat capacity at constant volume, and H_i is the total enthalpy. Because it is assumed the reactor experiences heat exchange with the environment, A is the heat exchange surface area and U is the overall heat transfer coefficient. Finally, R is the gas constant and C_{tot} is the total molar concentration of material in reactor. The key parameters which must be preset in equation 3.8 and 3.9 are T, C_i, U, A, V, Q_{in}, Q_{out}. Other variables not shown explicitly, which are important when calculating the total enthalpy of the inlet stream, are T_{in} P_{in} and, which are the temperature and pressure of the inlet stream.

Equation 3.8 is the energy balance for the CSTR reactor. Because ideal gases are considered, the ideal gas law which is

$$PV = nRT, \tag{3.10}$$

is used in the derivation to relate the current pressure P of gas to the volume V, the temperature T, and the number of moles n. The total enthalpy is a way to quantify the energy stored by each species. The heat capacity at constant volume C_v is also an important species-specific value which is a way to quantify the amount of energy stored by a species as a change in temperature. To calculate these terms, thermodynamic reference-state quantities must be known. The reference-state quantities from 7-parameter NASA polynomials are used, as well as the Maxwell relations, to derive other necessary thermodynamic properties from the equations of state [10].

Equation 3.8 defines the mass balance for the system as the rate of change in molar concentration for each species. The species-specific generation rate is the variable $\dot{w}_i$. In this model, the reactions are considered elementary, which means the instantaneous concentration of each species in the mixture governs this rate of reaction. The reactions used for hydrogen combustion are all considered reversible, which means conversion from reactants to products and products to reactants happens simultaneously.

For a system of M_i, $i = 1 ... n_s$ chemical species, the species react according to $k = 1 ... n_r$ reversible elementary reactions defined by

$$\sum_{i=1}^{n_s} v_{ik}^{'} M_i \rightleftharpoons \sum_{i=1}^{n_s} v_{ik}^{''} M_i, \tag{3.11}$$

where $v_{ik}^{'}$ and $v_{ik}^{"}$ are the stoichiometric coefficients of species i in reaction k. The rate of progress variables can be defined for every kth reversible reaction as [18] [21]

$$q_k = q_k^f - q_k^r = k_k^f \prod_{i=1}^{n_s} [X_i]^{v_{ik}^{'}} - k_k^r \prod_{i=1}^{ns} [Xi]^{v_{ik}^{"}}. \tag{3.12}$$

The parameters k_k^f and k_k^r are the forward and reverse rate constants. In this model, the forward rate constant is defined by the modified Arrhenius equation which is [22]

$$k_f = AT^b e^{\frac{-E_a}{RT}}, \tag{3.13}$$

The forward and backward rate constants are related through the equilibrium constant via

$$k_r = \frac{k_f}{K_c}. \tag{3.14}$$

For elementary reactions, as in the case here, the forward rate constant is a function of temperature, and in more complex cases, it can also depend upon other factors such as pressure.

Finally, the production rate of a single species i is calculated by summing up the production rate of progress variables for all reactions involving species i through

$$\dot{w}_i = \sum_{k=1}^{nr} v_{ik} q_k. \tag{3.15}$$

The Cantera Library [10] is used to generate and solve the governing system of equations 3.8, and 3.9. The library reads an input file containing the required thermodynamic reference data for all 11 species. The name of the yaml file is called *H2o2.yaml*. To model the CSTR, an `IdealGasReactor` object is created and combined with two `MassFlowController` objects to model the inlet and outlet flows. A `Wall` object is also added to model the heat exchange between the reactor and the surrounding oven environment.

The governing equations are used to model the transient startup of the CSTR. Before time starts, the reactor is filled up to the defined volume with the inlet gas mixture of hydrogen (H_2), oxygen (O_2), and nitrogen (N_2). Oxygen and hydrogen are required for ignition, and nitrogen is added to adjust the absolute partial pressures of both gases. When adjusting the mole fractions of each species in the inlet stream, only the mole fraction of hydrogen is changed for simplicity and the concentration of Oxygen is held constant. This is the same as using the concentration of hydrogen to adjust the air-to-fuel ratio, which is the critical parameter in combustion. Enough nitrogen is added then to make up the difference.

Cantera uses a slightly different version for the governing equations 3.8 and 3.9. Therefore, instead of using the molar concentration of each species $[X_i]$, the dimensionless molar fraction is used. Both represent a very similar quantity and can be converted through the relation

$$X_{ki} = \frac{[X_k]}{\sum_{i=1}^{K} [X_i]}. \tag{3.16}$$

Additionally, in this implementation, the mass flowrate $\dot{m_{in}}$ is defined instead of the volumetric flowrate Q_{in}. This means Q_{in} is equal to the mass flow rate $\dot{m_{in}}$ divided by the density of the inlet ρ_{in} and the outlet volumetric flow rateQ_{out} is equal to $\dot{m_{out}}$ divided by the density of the outlet ρ. Filling the reactor over time will not be modeled, so the mass flow rate out of the reactor is set equal to the mass flow rate into the reactor. In the case of a real CSTR, this would be equivalent to using two mass flow valves set to the same setting. Table 3.1 shows all the constant parameters used as input for the Cantera simulation.

Table 3.1: Parameters for the Transient Hydrogen Combustion CSTR System.

Parameter	Value	Units
$\dot{m}_{\text{in}}$	2.5×10^{-8}	kg/s
P_0	1330.3	Pa
U	5.0	W/m^2
A	0.2	m^2
V	0.0009	m^3
χ_{O2}^0	0.12	Unitless

From the values in table 3.1, the residence time of the CSTR is calculated to be about 200 seconds. When adjusting the composition of the inlet stream, only the mole fraction of hydrogen will be adjusted because the parameter that affects the combustion is the air-to-fuel ratio. The mole fraction of oxygen is always held at 0.12, and enough nitrogen is added to make up the difference.

To solve the system of equations, Cantera uses the CVODES solver, an efficient solver for stiff and non-stiff ODEs. CVODES is a part of the SUNDIALS suite created by the Lawrence Livermore National Laboratory. Cantera generates a Jacobian array for the governing ODEs, which, along with residual evaluation functions, are passed to the `Integrator` method, which uses CVODES. Because the governing equations are stiff, Backwards Differentiation Formula (BDF) in fixed-leading coefficient (FLC) form is used as the implicit integration method. At each integration step, a nonlinear system of equations must be solved using one of many nonlinear solver choices, such as the Newton method or a Krylov method depending upon the system's complexity. The user can utilize Preconditioners in Cantera for potentially faster integration, but in this case, they are not used. CVODES also uses adaptive step sizing by estimating the local error at each integration step and decreasing it if it does not satisfy tolerance conditions. When the time steps for integration are specified, CVODES autonomously chooses the step size required to reach the specified time steps but only reports the solution at the desired time step [13][10].

In this example, it is assumed the purpose of modeling the dynamic system is to better understand the behavior of the reactor during initial startup until the steady state operation is reached. The purpose of modeling could be, for example, for safety or sensitivity testing during the reactor design phase. The assumption of steady-state conditions in a CSTR greatly simplifies the governing equations, so much so that a satisfactory analytical solution is usually available. In contrast, the transient startup behavior of a CSTR is much more complicated and there is quite a bit of knowledge added value in modeling it.

The key variables that are tracked during the reactor start-up are reactor temperature T_r and reactor hydrogen mole fraction X_{H_2}. Only the inlet gas temperature T_{in} and the inlet mole fraction of hydrogen $X^0_{H_2}$ in the inlet stream are adjusted. The initial conditions for all other parameters are fixed. It should be noted that due to the details of the reactor setup, the reactor temperature T_r and hydrogen mole fraction X_{H2} will also be initialized the same as the inlet stream because, at the beginning of the numerical model, the reactor is filled to the specified reactor volume V with inlet gas.

The mole fractions of oxygen, water, and the intermediate species involved in the combustion process are also essential components of the governing system of equations 3.8 and 3.9 as they drive the rate of reaction. However, for this illustration, it is assumed that only hydrogen concentration and temperature are variables that can be monitored throughout reactor operation. This limitation is practical because, in real-world experiments, not all variables influencing a dynamic system may be controllable or even known. Chemical intermediates exist in unstable equilibrium and often form and dissipate so rapidly that measuring their concentration is unfeasible. Additionally, their introduction into the inlet gas stream is impractical, preventing their use for setting initial conditions.

Although the underlying dynamic system is of high dimensionality due to a large number of reacting species, from the perspective of using kernel EDMD-DL to reconstruct the system, there are only two. The other variables are lumped into the "nonlinearity" of the dynamics. For the kernel EDMD-ML method to be useful for predicting dynamic systems from experimental data, it must be robust to such situations.

3.3.4 Simultaneous Modeling of Temperature and Hydrogen Concentration Variations

Data collection

In the first attempt, kernel EDMD-DL with the RBF kernel is used to predict the dynamics of the transient CSTR startup behavior using initial conditions which vary in both the inlet temperature T_{in} and initial hydrogen mole fraction $X^0_{H_2}$. All other species' mole fractions are zero at $t = 0s$. 9 initial conditions are evenly distributed from $T_{in} = 775$ K to $T_{in} = 782$ K and $X^0_{H2} = 0.06$ to $X^0_{H2} = 0.08$. All of the temperature values are above the autoignition temperature of the system, which means that ignition can occur spontaneously.

Unlike the Hopf bifurcation defined by the solution to the cubic Lotka-Volterra equations in section 3.2, starting conditions for all state-space parameters cannot be defined over the ranges that they exist in during a trajectory as initial conditions. Therefore, each trajectory must be long enough to capture the duration of the startup dynamics. The number of initial conditions which can be tested is thus limited by computation time. In this case, the number of trajectories is set to 9, and the spacing between trajectories is chosen to be as close as possible without overlapping.

For all 9 trajectories, the governing equations are solved using the CVODES solver from $t = 0s$ to $t = 0.35s$. The numerical solutions can be viewed in the first row of figure 3.10. The top leftmost graph in the figure is a plot of reactor temperature T_r vs the reactor hydrogen mole fraction X_{H2}. Different colors represent trajectories with different initial conditions of varying inlet gas temperature and hydrogen concentration. The trajectories move from the right bottom corner of the graph to the left over time. The right-most graphs show X_{H2} and T_r vs time separately. The autoignition of each trajectory can clearly be seen in each graph by the sudden increase in temperature and sudden decrease in hydrogen mole fraction in the reactor. This time difference between heat application and ignition is called the ignition delay time. During the ignition delay, intermediate species are formed in chain reactions, eventually leading to the final stable reaction products and a propagating flame front [7]. Ignition delay times are an important aspect of combustion because they directly affect the efficiency and performance of combustion systems. For instance, a shorter ignition delay often leads to improved engine power output in internal combustion engines. Ignition delay time also affects the formation of pollutants during combustion, such as nitrogen oxides (NOx).

During the ignition, the temperature rises about 15 K (depending upon the starting temperature) and then decreases again as the heat energy is proportionally removed and a continuously burning flame stabilizes at a temperature similar to the surrounding oven temperature. This state is the steady state of the CSTR. If the numerical simulation were

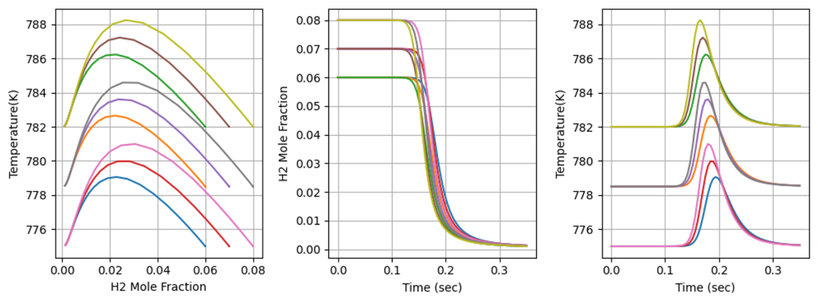
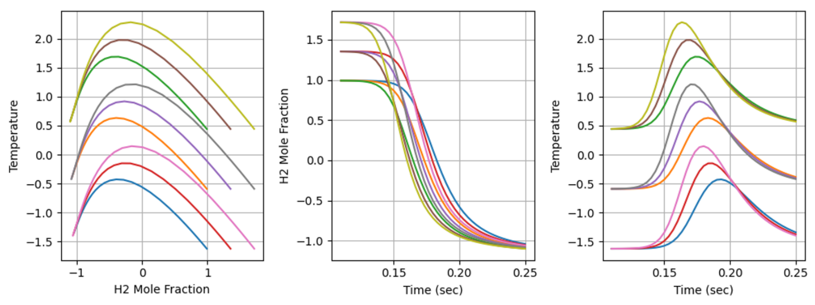
	Numerical Solution
Original Numerical Solution	
Cleaned Data from Numerical Solution (Normalized+trimmed)	

Figure 3.10: Numerical Solutions for hydrogen combustion in constant volume CSTR example for 9 initial conditions.

to be simulated to $t = \infty s$, theoretically, the reactor conditions would remain the same. This is another very important aspect of the model because the steady-state operating conditions of the CSTR directly affect the quality and quantity of the products. From the analysis of figure **??**, It appears that the autoignition time decreases as temperature and hydrogen mole fraction increase. This agrees with the literature [7].

It can be observed that there are some characteristics of the data that make it not ideal for use with the kernel EDMD-DL method or other machine learning methods in general. First, the scales of T_r and X_{H2} are drastically different from each other. This can be resolved by normalizing the data in each dimension. Here standard score normalization is used, which is

$$X = \frac{X - \mu}{\sigma}. \tag{3.17}$$

Equation 3.17 is used to normalize both X and Y matrices. Where X is either X or Y, μ is the mean of all the collected samples from all trajectories combined, and σ is the standard deviation of all the collected samples from all trajectories combined. After normalization, the data is still labeled as temperature and mole fraction to easily distinguish between the two dimensions but the units are removed for temperature because Kelvin is no longer accurate.

the units of Kelvin are no longer displayed in graphs of the data, however, the axis is still labeled as Temperature.

An additional problem with the data is that before and after ignition, there are regions of very slow movement in the dynamic variables as compared to the fast dynamics during ignition. These multiscale dynamics can lead to a poor approximation of the Koopman operator because important fast dynamics may be lost when a lower dimensional system approximates the model. Additionally, it could lead to an operator that is ill-conditioned. One option to mitigate this problem is to apply another function to the data to transform it in a way that smooths out the dynamics.

The data is not transformed for the reaction system modeled here, but the trajectory time is truncated to $t = 0.1s$ to $t = 0.25s$. This removes a large portion of the slow dynamics from the sampled data. In addition to reducing the impact of slow dynamics on the calculation of the approximated koopman operator, it also increases the resolution of the data if the number of timesteps remains the same, hopefully leading to improved prediction capabilities of the fast dynamics during ignition. One drawback to this strategy is that important portions of the system's dynamic behavior will not be modeled. A way to get around this would be to multiple times on different regions of the data but this can quickly become computationally expensive.

Hyperparameter Tuning and Results

After truncating and normalizing the data as described in the previous section, 40 evenly distributed data values corresponding to 40 time steps are selected from each trajectory to calculate the approximated Koopman operator. 40 steps are chosen rather arbitrarily, mostly taking into account computation time. The number of time steps is increased from the Hopf bifurcation example, however, in an attempt to increase resolution and reduce the effect of the multiscale dynamics. Because there are fewer overall trajectories in the dataset as compared to the Hopf bifurcation example, however, the computation time is comparable.

As was done for the Hopf system, a grid search is performed over various θ values and the number of eigenfunctions and the OOS error and in-sample loss are calculated for each parameter combination. Figure 3.11 shows the results.

Figure 3.11 exhibits a visually interesting pattern. Generally, for the in-sample loss, the optimal corresponding θ value decreases as the number of eigenfunctions increases. The average OOS trajectory error shows that the error increases as both the number of eigenfunctions and the value for theta increase. The region of overfitting is clearly in the right-hand upper corner of both graphs. It appears there might be a larger potential for agreement between the two accuracy measures when a smaller number of eigenfunctions is used. One noticeable feature is that on average, the oos error values are much higher than in the grid search performed for the Hopf example in figure 3.5. Additionally, they are much higher than average in-sample loss values. This big difference in the scales of

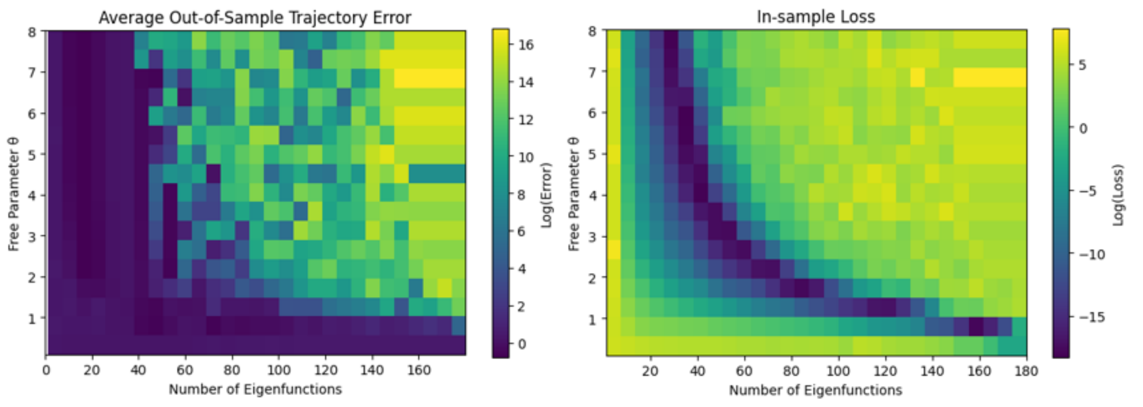


Figure 3.11: In-sample loss and OOS trajectory error using RBF Kernel for different combinations of the free parameter θ and eigenfunctions for hydrogen combustion example.

the data is likely the cause of a bad prediction for the optimum by the combined error function.

From Figure 3.11, it is evident that finding a parameter combination that achieves accurate predictions for both the out-of-sample (OOS) Error and in-sample loss is challenging. The combined error equation 3.5is at a minimum at 10 eigenfunctions and a θ parameter value of 3.0. Figure 3.12 shows a more detailed view of the loss landscape at loss landscape for 10 eigenfunctions.

Learning is performed using the RMSprop optimizer to enhance training in a rugged loss landscape, as described in Section 3.2.4 (data not shown here). After 500 Epochs, this results in an optimal θ parameter of 17.9, which by observing figure 3.12 it is not the global minimum. When a θ value of 17.9 and 10 eigenfunctions are used to reconstruct the dynamics, the OOS trajectory error is extremely high, indicating a poor fit. It is therefore necessary to admit the failure of the hyperparameter search strategy and use a manual guess-and-check procedure.

Even after a manual parameter search was exhaustively undertaken, none of the combinations tested produced predictions comparable accuracy to the reportings from the Hopf bifurcation example. Despite this, however, The best results of the search are still reported in figure 3.13.

Considering the poor accuracy values reported in 3.13, one can be certain the kernel EDMD-DL method with the RBF kernel is not very successful in reconstructing the combustion dynamics. Because a very intensive hyperparameter search is performed, it is less likely to be due to an inadequate hyperparameter combination than it is to be due to the characteristics of the data used for training.

A simple sensitivity analysis is conducted to try to understand what about the data that is causing the low accuracy. For each new condition tested, new hyperparameter values

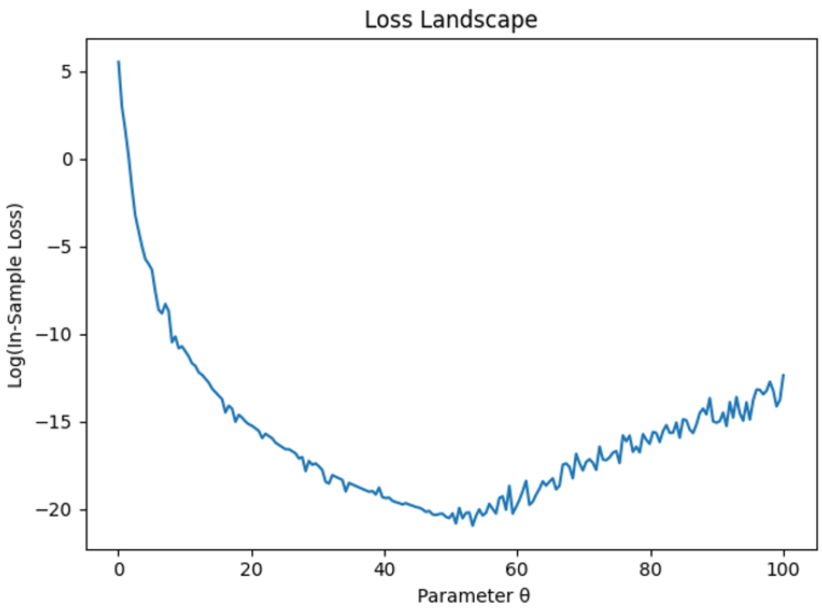


Figure 3.12: In-Sample Loss vs. Free Parameter θ for Hopf bifurcation data with 10 eigenfunctions and 40 time steps for the hydrogen combustion example

are needed to find an optimized prediction. This causes a major source of variation and decreases the reliability of the results from the sensitivity analysis; however, due to the thoroughness of the parameter search, the results are considered reliable enough to make generalized conclusions.

First, an increase in the number of time steps is tested. This is concluded not to have an effect because grid searches using anywhere from 100 to 200 time steps yield similar results and do not significantly improve the accuracy. Over 200 time steps became time prohibitive in terms of computational time. In a second attempt, the time range of the data samples is decreased from $t = 0.1s$ to $t = 0.25s$ to $t = 0.125s$ to $t = 0.23s$ to limit as much as possible the effects of the multiscale dynamics but still preserve the essential dynamics of the system. As in the first case, this also does not improve the accuracy of the results significantly. Finally, the range of values for both the mole fraction and the temperature is adjusted to test the idea that maybe the approximated Koopman operator does not have enough knowledge about the complexity of the dynamics between sample values or maybe the trajectories are too close to each other causing amplification of numerical errors. This last attempt, however, also did not improve the accuracy. Increasing and decreasing the space between the initial condition both decreased the accuracy.

A close look at the underlying dynamic system shows that the dynamic behavior of trajectories with very similar initial conditions can show drastically different behavior over time. This is due to the high dimensionality and complexity of the underlying dynamics.

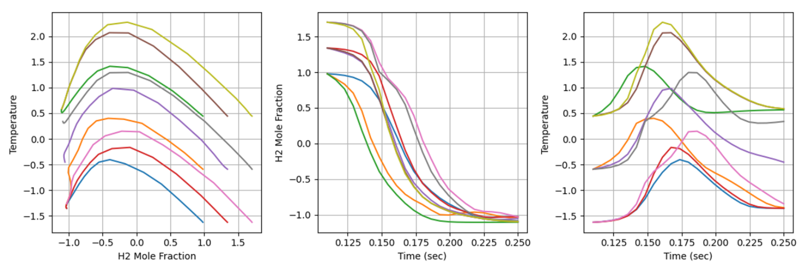
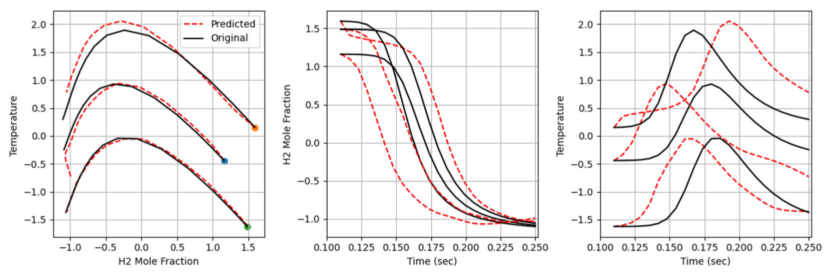
RBF Kernel	
In-Sample Predictions	
Out-of-Sample Trajectories Initial conditions [0.065,779] , [0.077,781] , [0.074,775]	
Hyperparameters	Time Start = 0 sec Time End = 0.25 sec Sample time range for kernel EDMD-DL = 0.1-0.25 sec Number of Timesteps = 40 Number Eigenfunctions = 60 Free parameter ϴ =0.5
Accuracy	In-Sample Loss = 0.900 Average OOS Error =0.8211

Figure 3.13: In-Sample Loss, OOS Trajectories, accuracy reports, and hyperparameter values for Combustion data example using RBF kernel and varying both inlet temperature and hydrogen mole fraction.

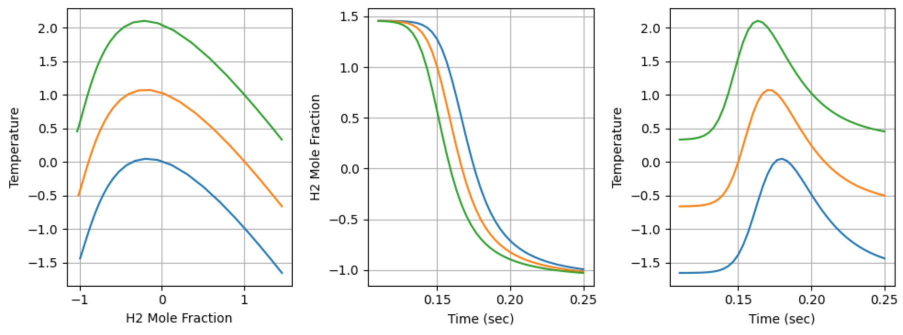
Additionally, in figure 3.10, it can be observed the observed trajectories overlap each other at the end of the ignition as the steady state of the CSTR is reached. This means that there are not enough time-dependent variables tracked (the dimensionality of the state space vector is not high enough) for the approximated Koopman operator to adequately distinguish a unique path for each trajectory. This effect is most prominent in the hydrogen mole fraction dimension because, despite the starting mole fraction, all reactors experience nearly complete conversion and, in the end, reach the same steady state because of the long reactor residence time. In the next section, section 3.3.5, the reactor temperature and hydrogen mole fraction are modeled separately using kernel EDMD-ML to see if the accuracy of the results can be improved.

As a side note, a final reason for the inaccurate prediction results could also be that the RBF basis functions are inadequate to represent the system. This idea is not tested, however, due to time constraints.

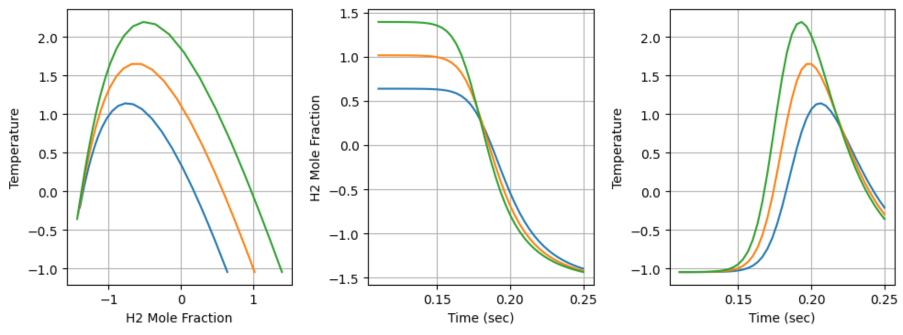
3.3.5 Separately Modeling Temperature and Hydrogen Concentration Variations

Data collection

Based on the findings in section 3.3.5, this time, data is numerically generated for initial conditions that only vary in temperature and hydrogen mole fraction separately. One data set contains the numerical solution for 3 initial conditions evenly distributed from $T_{in} = 775$ K to $T_{in} = 782$ K with $X^0_{H2} = 0.08$. This is referred to as the temperature data set. The second data set contains the numerical solution for 3 initial conditions evenly distributed from $X^0_{H2} = 0.06$ to $X^0_{H2} = 0.08$ at $T_{in} = 770$ K. This dat is referred to as the mole fraction dataset. The mole fraction and temperature ranges are not altered to keep comparability to the previous section. As before, this data is normalized and truncated from $t = 0.1s$ to $t = 0.25s$. The cleaned data can be viewed in figures 3.14a and 3.14b.



(a) Numerical solution of the combustion reaction varying temperature in the inlet gas and keeping hydrogen mole fraction the same.



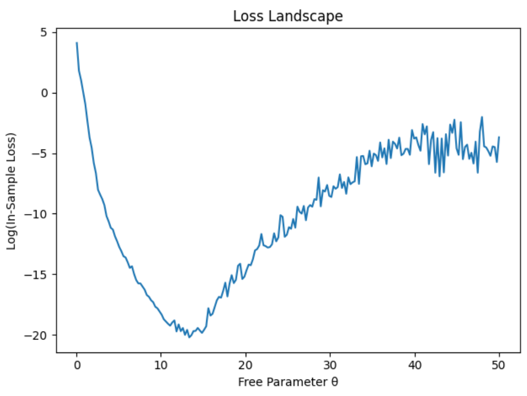
(b) Numerical solution of the combustion reaction varying hydrogen mole fraction in the inlet gas and keeping the inlet temperature the same.

Figure 3.14: Numerical solutions of the combustion reaction for 3 initial points varying either hydrogen mole fraction or temperature in the inlet gas.

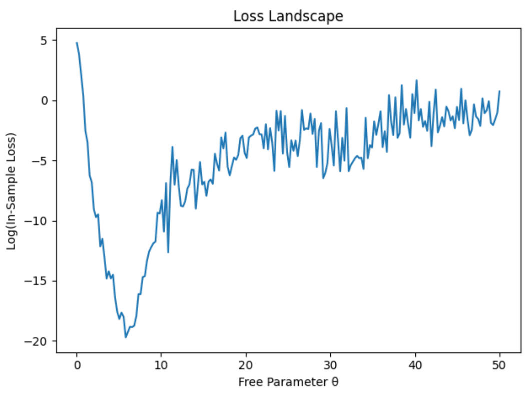
Hyperparameter Tuning and Results

Next, two separate grid searches are performed for each data set to find an adequate initial condition for the free parameter θ and the number of eigenfunctions, as in the example from the previous section 3.3.4 the number of time steps is set to 40 to maintain consistency.

The findings from these searches reveal that for the temperature data set, the combined minimum error is lowest at a θ value of 3.0 and 15 eigenfunctions (grid search results not shown). For the mole fraction dataset, the combined minimum error is lowest at a θ value of 2.5 and 20 eigenfunctions (grid search results not shown). The in-sample loss landscapes for both datasets are explored, keeping the number of eigenfunctions constant. Figure 3.15 shows the results for both datasets. An interesting observation that can be made about the in-sample loss data from both graphs is that the temperature data has a bigger global minimum in-sample loss, and it is smoother, for most θ values tested.



(a) In-Sample Loss vs. θ varying only inlet gas temperature in the initial conditions. 15 eigenfunctions and 40 time steps are used.



(b) In-sample Loss vs. θ varying only hydrogen mole fraction in the initial conditions. 20 eigenfunctions and 40 time steps are used.

Figure 3.15: Comparison of in-sample loss vs. Free Parameter θ for the combustion reaction example.

The gradient learning method is then applied to both data sets using the RMSprop optimizer to minimize the in-sample loss by optimizing the θ parameter. The results of this are 8.943 for the hydrogen mole fraction data and 11.034 for the temperature data set. Finally, because the parameter combinations prove to be successful providing adequate OOS error and in-sample loss, the hyperparameters are fine-tuned by hand to maximize the in-sample loss. The in-sample loss, OOS trajectories, accuracy reports, and final hyperparameter values can be viewed in figures 3.16 and 3.17.

The results displayed in figures 3.16 and 3.17 are more accurate than for the dataset varying both temperature and hydrogen mole fraction. Additionally, hyperparameter tuning is much quicker and easier for both because there is a larger region of agreement between

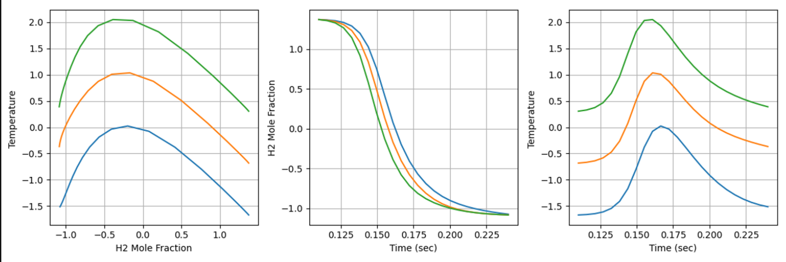
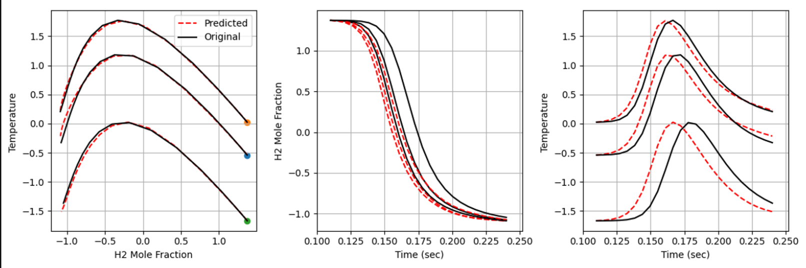
RBF Kernel	
In-Sample Predictions	
Out-of-Sample Trajectories Initial conditions: [0.08,779] , [0.08,781] , [0.08,776]	
Hyperparameters	Time Start = 0 sec Time End = 0.25 sec Sample time range for kernel EDMD-DL = 0.1-0.25 sec Number of Timesteps = 40, 138* Number Eigenfunctions = 17 Free parameter Θ =10.0
Accuracy	In-Sample Loss = 2.259e-09 Average OOS Error =0.299, 0.183* *OOS error decreases when time step size is increased to 138

Figure 3.16: In-Sample Loss, OOS Trajectories, accuracy reports, and hyperparameter values for combustion data example using RBF kernel and varying only inlet temperature while keeping hydrogen mole fraction the same.

the OOS trajectory errors and the in-sample losses. This phenomenon supports the hypothesis described in the previous section 3.3.4 that the data set is not well suited for the kernel EDMD-DL method. Separating the data into two sets gives the method a simpler dynamic system to recreate. The hyperparameters for the temperature and hydrogen mole fraction data sets are different from each other, indicating that it is difficult to capture the dynamics in both dimensions using the same reduced model.

Compared to the hydrogen mole fraction data, the temperature data set performed better for both the in-sample loss and OOS errors. The reason for this is likely due to the clear separation between observed trajectories, the dynamics are smoother, and compared to all other dimensions, the gas inlet temperature has one of the biggest effects on reactor temperature and therefore captures the majority of the change in the dynamics. It might be possible to improve the prediction of the hydrogen mole fraction data set if the mole fractions of more species are tracked. It could also be that the multiscale dynamics are too extreme for the approximated Koopman operator to reconstruct. If this is the case, try-

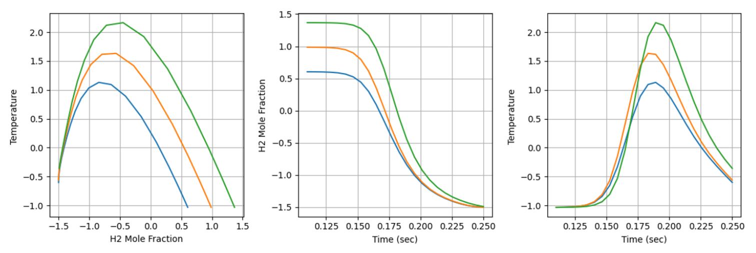
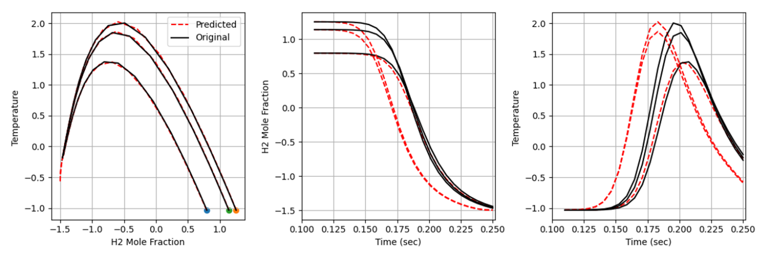
RBF Kernel	
In-Sample Predictions	
Out-of-Sample Trajectories Initial conditions [0.065,770] , [0.074,770] , [0.077,770]	
Hyperparameters	Time Start = 0 sec Time End = 0.25 sec Sample time range for kernel EDMD-DL = 0.1-0.25 sec Number of Timesteps = 40 Number Eigenfunctions = 20 Free parameter Θ =7.0
Accuracy	In-Sample Loss = 1.9168e-07 Average OOS Error =0.591

Figure 3.17: In-Sample Loss, OOS Trajectories, accuracy reports, and hyperparamter values for Combustion data example using RBF kernel and varying only inlet hydrogen mole fraction while keeping temperature the same.

ing out different types of basis functions might be useful rather than just the RBF kernel because potentially another set of basis functions is better suited. Additionally, it may be possible to break the data up into smaller time frames and model the dynamics of each section separately. In the end, it could be possible to piece the models together to recreate the entire trajectory. Unfortunately, none of these ideas are tested in this thesis.

Modeling an Expanded Range of Temperature Variations

To better understand the prediction capabilities of the kernel EDMD- method just for just predicting changes in temperature, the desired range of inlet gas temperatures is expanded to $T_{in} = 760$ K to $T_{in} = 790$ K and $X^0_{H2} = 0.08$. This time, 7 different initial conditions are used to numerically generate trajectories equally spread across the operating temperatures. The number of initial conditions is increased to test the hypothesis that increasing the number of trajectories to train the model increases the accuracy. The numerically generated data after normalization and truncation can be viewed in figure 3.18.

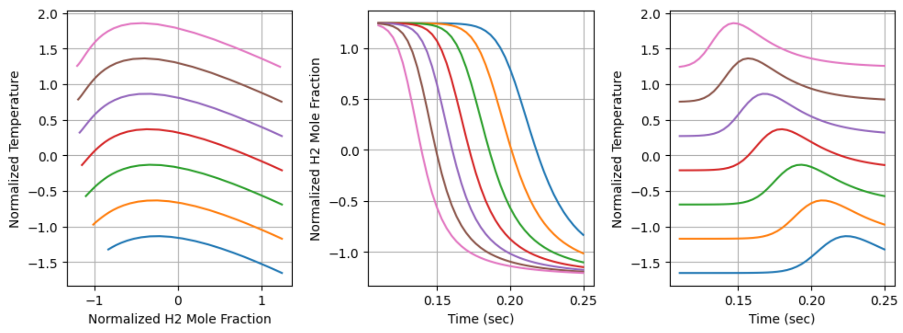


Figure 3.18: Numerical solution of the combustion reaction for 7 initial points varying an expanded range of inlet gas temperatures and keeping the inlet hydrogen mole fraction the same.

The same procedure is used for hyperparameter search and free parameter θ learning. As a result of the grid search (results not shown), the minimum combination error occurs at a θ value of 8.0 and 25 eigenfunctions. After adjusting the learning rate, the gradient learning method found an optimal value of θ at 4.305 with a learning rate of $1e-2$ and an initial value of 8.0. The most optimal final results for minimizing both the OOS error and in-sample loss are displayed in figure 3.19 The results of modeling the expanded temperature range are surprisingly accurate . The reported OOS trajectories and the in-sample errors are similar to those reported for the predictions of the Hopf bifurcation in section 3.2.2 for the polynomial and RBF kernels. Additionally, the accuracy values are nearly comparable to those reported literature for better-known and lower dimensional systems, often with known analytical solutions [21] [1]. However, the latter statement must be taken lightly due to the varying types of accuracy measures. Compared to other examples of EDMD applications in literature [21][1][31][20], the number of trajectories within the state-space used for training is quite small due to computational time limits. It would be interesting to see how the prediction accuracy could improve if the kernel EDMD-DL code is better optimized for efficiency.

3.3.6 Modeling a Second Dynamic Reaction System

Hydrogen combustion dynamics can exhibit a variety of interesting bifurcations [7]. In this last section, another set of solutions to the governing equations is explored. This time with the settings in table 3.2.

Compared to the previous reactor system, this system has a smaller starting pressure, a higher heat removal rate, a larger inlet gas flow rate, and a smaller volume. All factors contribute to a smaller ignition event and a lower conversion of reactants to products. The temperature in the reactor increases less during ignition, and the hydrogen mole fraction in the reactor decreases but not as much as in the first example. Additionally, the reactor

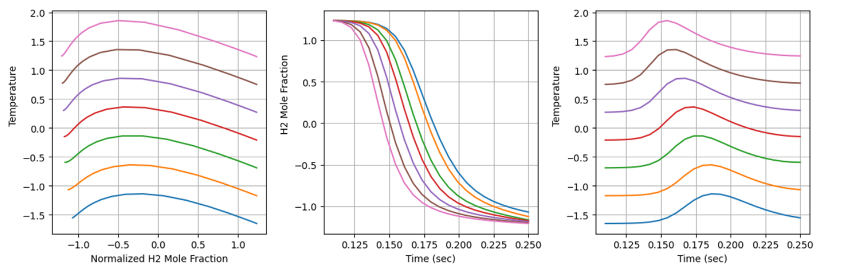
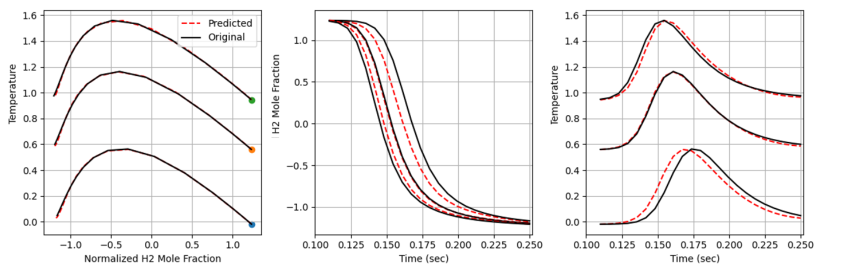
RBF Kernel	
In-Sample Predictions	
Out-of-Sample Trajectories Initial conditions [0.08,777] , [0.08,783] , [0.08,787]	
Hyperparameters	Time Start = 0 sec Time End = 0.25 sec Sample time range for kernel EDMD-DL = 0.1-0.25 sec Number of Timesteps = 40 Number Eigenfunctions = 24 Free parameter Θ =8.5
Accuracy	In-Sample Loss = 1.1873e-07 Average OOS Error =0.089

Figure 3.19: In-Sample Loss, OOS Trajectories, accuracy reports, and hyperparameter values for hydrogen combustion example using RBF kernel and varying only inlet temperature while inlet hydrogen mole fraction remains the same for an expanded range of temperature values

volume is much smaller, so there is less material in the reactor at one time to absorb the heat energy released during combustion. These factors result in a reactor system with a steady state higher than the inlet gas and less extreme autoignition. From an efficiency perspective, these reactor conditions are not ideal because fewer reactants are converted to products. The upside, however, is that this system is potentially more environmentally friendly because milder autoignition conditions can reduce the risk of side reactions that cause pollutants.

The new reaction system can be viewed in figure 3.20 with 9 different initial conditions evenly spaced between inlet gas temperature values of 776 K and 792 K and a constant value of 0.06 hydrogen mole fraction. The top graph is the original numerical solution from $t = 0s$ to $t = 0.3s$. The bottom graph is the normalized data which is trimmed from

Table 3.2: Parameters for the 2nd example of the Transient Hydrogen Combustion CSTR System.

Parameter	Value	Units
$\dot{m}_{\text{in}}$	70.5×10^{-8}	kg/s
P_0	1199.7	Pa
U	2.0	W/m^2
A	0.2	m^2
V	5×10^{-6}	m^3

$t = 0.1s$ to $t = 0.25s$. In the figure, one can clearly see how each trajectory (represented by a different color) converges into a steady state with a higher temperature than the starting temperature at the end of modeling time. As compared to the data from the first example, this data is smoother and the trajectories are more uniform across temperatures. As with the other examples, a grid search is conducted (results not shown), holding the number of time steps constant at 40. The combined error is pointed to be lowest at a θ value of 2.0 and 42 eigenfunctions. This time, parameter learning for θ is skipped, and only guess-and-check adjustments are made to theta and the number of eigenfunctions to fine-tune the results. Surprisingly very few adjustments need to be made. The results of this parameter search and using the kernel EDMD-DL method to model this new reaction system are presented in Figure 3.21. Compared to the first reactor example where only varying temperature is modeled in section 3.3.5, the accuracy of the predictions from the kernel EDMD-DL method applied to the second reactor example performed equally as well. This indicates that the kernel EDMD-DL method can model different combustion bifurcations as long as the data is presented properly to the method.

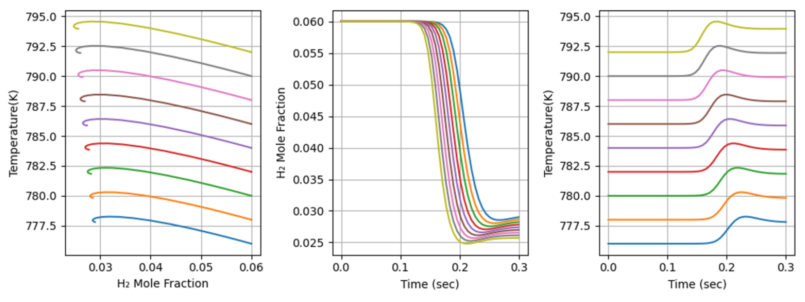
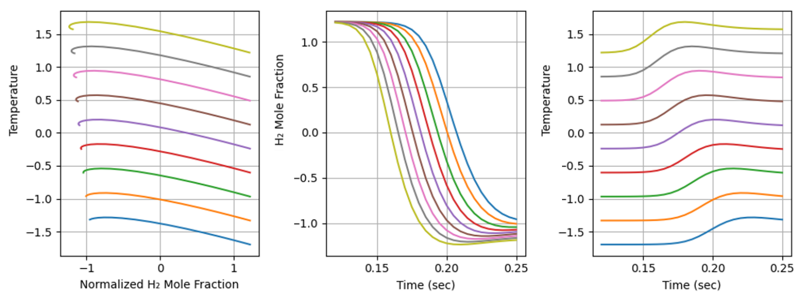
	Numerical Solution
Original Numerical Solution	
Cleaned Data from Numerical Solution (Normalized+trimmed)	

Figure 3.20: Numerical solution of the second combustion reactor system example for 9 initial points only varying in inlet gas temperature

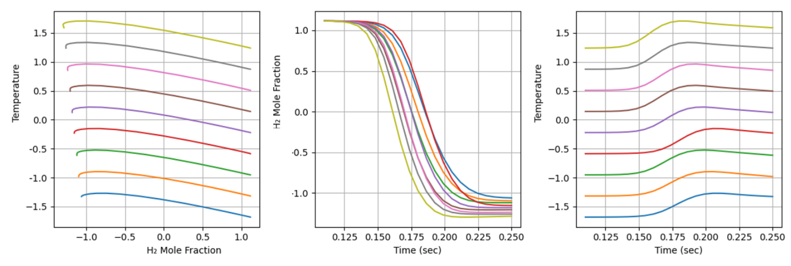
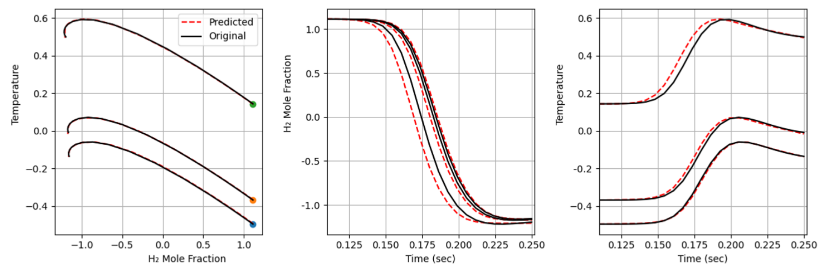
RBF Kernel	
In-Sample Predictions	
Out-of-Sample Trajectories Initial conditions [0.06,782.5] , [0.06,783.2] , [0.06,786]	
Hyperparameters	Time Start = 0 sec Time End = 0.3 sec Sample time range for kernel EDMD-DL = 0.1-0.25 sec Number of Timesteps = 40 Number Eigenfunctions = 42 Free parameter Θ =3.0
Accuracy	In-Sample Loss = 1.27e-07 Average OOS Error =0.0732

Figure 3.21: In-Sample Loss, OOS Trajectories, accuracy reports, and hyperparamter values for combustion data modeling the second combustion reactor system example with 9 initial conditions only varying in inlet gas temperature.

4 Conclusion

4.1 Summary

This thesis discusses the mathematical theory behind kernel-based EDMD and Kernel EDMD-DL. These theories are then applied to predict the behavior of nonlinear dynamical systems using an approximated linear operator from numerically generated data. The resulting accuracy of the predictions is assessed. Additionally, the capabilities and applications of the gradient kernel parameter learning in the Kernel EDMD-DL method are analyzed in detail. The kernel EDMD and kernel EDMD-DL methods are applied successfully applied to the cubic Lotka-Volterra equations with a Hopf bifurcation as a proof of concept. As a next step, the kernel EDMD-DL method is applied to predict the dynamics of hydrogen combustion in a constant volume transient CSTR. In this case, the kernel EDMD-DL method is conditionally successful upon using quality input data and simplifying the dynamics by choosing a more limited parameter space. A highlighted component of this work is its focus on utilizing the kernel EDMD methods for making out-of-sample predictions and optimizing the required method hyperparameters to do so.

4.2 Discussion and Outlook

The results show that kernel EDMD-DL can be successfully applied to predict complex nonlinear dynamics. However, the practical implementation of the method is not as straightforward as one would initially think. One of the most important observations made is how critical it is to find optimal hyperparameters. In the case of the hydrogen combustion dynamics example, the accuracy of the results is extremely sensitive to the parameter values. A rather unexpected aspect is how easy it is to overfit the data. At first, it was expected that overfitting would not be a problem; however, this proved not to be the case.

By far, the most critical hyperparameter is the number of eigenfunctions. At its core, EDMD is a model order reduction technique, and the great sensitivity of the results to the number of eigenfunctions used to reconstruct the data highlighted this aspect. It is unfortunate that more attention is not paid to creating a more robust and effective method of assessing the quality of the eigenfunctions, as is done in other literature sources, for example, in [21]. A separate thesis could be written about analyzing the eigendecomposition of the approximated Koopman operator.

A grid search method is useful for parameter optimization but is not very computationally efficient. The effectiveness of the kernel EDMD-ML method could be drastically improved with a better way to optimize the parameters, which cannot be optimized using a gradient method. The combined error equation 3.5 used to assess both the out-of-sample error and in-sample error is only partially effective at finding a parameter combination that finds a minimum that appropriately favors the OOS trajectory error. One would think it might be enough to minimize the out-of-sample error; however, an analysis of the loss landscape for all examples shows that both parameters need to be optimized for the full parameter region of interest to be accurately reconstructed. In future work, exploring better concurrent optimization methods would be interesting.

The kernel learning method is shown to be contingently effective in finding an optimal kernel-free parameter through a gradient learning method. Theoretically, it works, but practically, it is difficult to implement. The loss landscape for both the Hopf and the combustion reaction examples shows many local minima, which makes learning challenging when it comes to both setting the initial value and choosing an appropriate optimizer. However, it cannot be concluded whether these issues are truly prohibitive to its usage in other applications. In the context of this thesis, the kernel learning method is not useful because, as discussed in the previous paragraph, it is not enough to minimize only the in-sample loss. Implementing a way to assess the OOS trajectory loss into the learning method would be a great next step.

Another critical observation made during the thesis work is the importance of using quality data to produce quality results. This is true for the kernel EDMD-DL method and all machine-learning applications. In the hydrogen combustion example, the data quality is a problem and prohibits the method's effectiveness. It is not completely understood if all challenges can be effectively worked around or if there are limitations. In the hydrogen combustion example, a workaround method is implemented by creating a separate approximated Koopman operator for the transient behavior of each system state. This is just one example of many different methods that can be used. It is worthwhile to conduct further research on effectively utilizing imperfect data.

As a final note, kernel EDMD holds a lot of promise for the future of dynamic system modeling. The drive to further integrate simulation and modeling into engineering and scientific applications is increasing, and kernel EDMD will be a significant contributor. Additionally, the challenges encountered with kernel parameter learning do not imply that the method cannot be effectively utilized. It is unfortunate that the beautiful theory behind the method can be obscured by the practical application difficulties and common machine learning challenges, but as demonstrated in this thesis, these obstacles can be overcome.

Bibliography

[1] Peter J. Baddoo, Benjamin Herrmann, Beverley J. McKeon, and Steven L. Brunton. Kernel learning for robust dynamic mode decomposition: linear and nonlinear disambiguation optimization. *Proceedings. Mathematical, physical, and engineering sciences*, 478(2260):20210830, 2022.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. Computer science. Springer, New York, NY, 2006.

[3] Erik Bolager. ”kernel edmd learning”, 2023.

[4] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2):e0150171, 2016.

[5] Steven L. Brunton and Jose Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, 2019.

[6] Marko Budisić, Ryan Mohr, and Igor Mezić. Applied koopmanism. *Chaos (Woodbury, N.Y.)*, 22(4):047510, 2012.

[7] F. P. Di Maio, G. Barbieri, and P. G. Lignola. Combustion processes in cstr. bifurcation analysis of the h 2 –o 2 system. *J. Chem. Soc., Faraday Trans.*, 92(16):2989–2996, 1996.

[8] Felix Dietrich, Thomas N. Thiem, and Ioannis G. Kevrekidis. On the koopman operator of algorithms. *SIAM Journal on Applied Dynamical Systems*, 19(2):860–885, 2020.

[9] Gary Froyland, Georg A. Gottwald, and Andy Hammerlindl. A computational method to extract macroscopic variables and their dynamics in multiscale systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1816–1846, 2014.

[10] David G. Goodwin, Harry K. Moffat, Ingmar Schoegl, Raymond L. Speth, and Bryan W. Weber. Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes. https://www.cantera.org, 2022. Version 2.6.0.

[11] Gregory P. Smith, David M. Golden, Michael Frenklach, Nigel W. Moriarty, Boris Eiteneer, Mikhail Goldenberg, C. Thomas Bowman, Ronald K. Hanson, Soonho

Song, William C. Gardiner, Jr., Vitali V. Lissianski, and Zhiwei Qin. Gri-mech 3.0, 10/30/2002.

[12] E. Hairer, S. P. Nørsett, and Gerhard Wanner. *Solving ordinary differential equations*, volume 8. Springer-Verlag, Berlin and New York, 2nd ed. edition, 1993-.

[13] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[14] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[15] Bingni W. Brunton J. Nathan Kutz, Steven L. Brunton and Joshua L. Proctor. *Chapter 1: Dynamic Mode Decomposition: An Introduction*, pages 1–24.

[16] A. B. Katok, Boris Hasselblatt, and Leonardo Mendoza. *Introduction to the modern theory of dynamical systems*, volume v.54 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, Cambridge, 1995.

[17] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, 1931.

[18] Mahdi Kooshkbaghi, Christos E. Frouzakis, Konstantinos Boulouchos, and Iliya V. Karlin. n-heptane/air combustion in perfectly stirred reactors: Dynamics, bifurcations and dominant reactions at critical conditions. *Combustion and Flame*, 162(9):3166–3179, 2015.

[19] Milan Korda and Igor Mezić. On convergence of extended dynamic mode decomposition to the koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

[20] Daniel Lehmberg, Felix Dietrich, Gerta Köster, and Hans-Joachim Bungartz. datafold: data-driven models for point clouds and time series on manifolds. *Journal of Open Source Software*, 5(51):2283, 2020.

[21] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos (Woodbury, N.Y.)*, 27(10):103111, 2017.

[22] Jorge Mario Marchetti. *Reaction engineering, catalyst preparation, and kinetics*. CRC Press, Boca Raton, 1st edition, 2021.

[23] Carsten Olm, István Gy. Zsély, Róbert Pálvölgyi, Tamás Varga, Tibor Nagy, Henry J. Curran, and Tamás Turányi. Comparison of the performance of several recent hydrogen combustion mechanisms. *Combustion and Flame*, 161(9):2219–2234, 2014.

[24] Taisuke Otsu. Matrix algebra, by karim m. abadir and jan r. magnus, cambridge university press, 2005. *Econometric Theory*, 22(05), 2006.

[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[26] Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*, volume 10 of *Springer eBook Collection Mathematics and Statistics*. Springer, New York, NY, 1999.

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

[28] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[29] Sebastian Verhelst and Thomas Wallner. Hydrogen-fueled internal combustion engines. *Progress in Energy and Combustion Science*, 35(6):490–527, 2009.

[30] Christoph Wehmeyer and Frank Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics.

[31] 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.

[32] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based approach to data-driven koopman spectral analysis.