UNIVERSITY OF CALIFORNIA SAN DIEGO

Data-Driven System Analysis Using the Koopman Operator: Eigenfunctions, Invariant Subspaces, and Accuracy Bounds

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Engineering Sciences (Mechanical Engineering)

by

Masih Haseli

2022

University of California San Diego

2022

DEDICATION

To pioneers who challenge the status quo and seek a better world.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This dissertation is the culmination of the efforts made by all the people I have been in contact with (possibly indirectly) in my lifetime. I have learned from many to reach this stage of my life. Although infeasible to thank everyone, I mention a few directly impacting this dissertation.

First, I would like to express my deepest gratitude to my advisor, Prof. Jorge Cortés, for his profound impact on my personal and professional life. His patience, integrity, objective thinking, and kindness have reshaped my attitude towards research and personal development. I sincerely thank Jorge for patiently helping me to become a better person. Looking back, I do not think I could find a better advisor. This dissertation is his more than it is mine.

I would like to thank my dissertation committee, Prof. Nikolay Atanasov, Prof. Boris Krämer, Prof. Miroslav Krstić, and Prof. Melvin Leok, for their time and effort. I sincerely thank them for providing me with guidance and constructive criticism. Having them on my committee was a great honor for me.

My time at UCSD was fruitful. During this time, I got the chance to learn from the best. I would like to express my gratitude to Prof. Robert Bitmead, Prof. Philip Gill, Prof. Miroslav Krstić, Prof. Maurício de Oliveira, Prof. Sonia Martinez, and Prof. Jorge Cortés from whom I have learned important lessons. Moreover, I would like to thank Behrooz, Vishaal, Pio, Amit, and Sven, whose role as teaching assistants was crucial.

I would like to express my gratitude to our current and former group members for creating a friendly and dynamic environment. Special thanks to Prof. Sonia Martinez and my advisor, Prof. Jorge Cortés, for organizing our weekly group meetings to discuss the latest advances in the field. Moreover, I would like to thank my precious friend and former office mate (and now professor), Erfan Nozari, from whom I have learned crucial lessons. I also would like to thank our former group member, Dariush Fooladivanda, for assisting me during a challenging time.

My utmost gratitude goes to my father for his unconditional love and support throughout my life. Particularly for believing in me and providing a safe and pleasant environment for me to grow. I will never forget his sacrifices.

The research reported in this dissertation was supported in part by the Office of Naval Research Award N00014-18-1-2828 and the National Science Foundation Award IIS-2007141.

Chapter 3, in part, is a reprint of the material [HC22a] as it appears in 'Learning Koopman Eigenfunctions and Invariant Subspaces from Data: Symmetric Subspace Decomposition' by M. Haseli and J. Cortés, in IEEE Transactions on Automatic Control 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material [HC21b] as it appears in 'Parallel Learning of Koopman Eigenfunctions and Invariant Subspaces For Accurate Long-Term Prediction' by M. Haseli and J. Cortés, in IEEE Transactions on Control of Network Systems 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 5 is taken, in part, from the material [HC21a] which has been submitted for publication as 'Generalizing dynamic mode decomposition: balancing accuracy and expressiveness in Koopman approximations' by M. Haseli and J. Cortés, in Automatica. The dissertation author was the primary investigator and author of this paper.

Chapter 6 is taken, in part, from the material [HC22b] which has been submitted for publication as ‘Temporal Forward-Backward Consistency, Not Residual Error, Measures the Prediction Accuracy of Extended Dynamic Mode Decomposition’ by M. Haseli and J. Cortés, in IEEE Control Systems Letters. The dissertation author was the primary investigator and author of this paper.

VITA

2013	Bachelor's Degree in Electrical Engineering – Control, Amirkabir University of Technology
2015	Master's Degree in Electrical Engineering – Control, Amirkabir University of Technology
2022	Doctor of Philosophy in Engineering Sciences (Mechanical Engineering), University of California San Diego

PUBLICATIONS

Journal publications:

[1] M. Haseli and J. Cortés, "Temporal Forward-Backward Consistency, Not Residual Error, Measures the Prediction Accuracy of Extended Dynamic Mode Decomposition," *IEEE Control Systems Letters*, 2022, submitted.

[2] M. Haseli and J. Cortés, "Generalizing Dynamic Mode Decomposition: Balancing Accuracy and Expressiveness in Koopman Approximations," *Automatica*, 2021, submitted.

[3] M. Haseli and J. Cortés, "Parallel learning of Koopman eigenfunctions and invariant subspaces for accurate long-term prediction," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 4, pp. 1833–1845, 2021.

[4] M. Haseli and J. Cortés, "Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric Subspace Decomposition," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3442–3457, 2022.

Conference proceedings:

[5] M. Haseli and J. Cortés, "Temporal Forward-Backward Consistency, Not Residual Error, Measures the Prediction Accuracy of Extended Dynamic Mode Decomposition," in *American Control Conference*, San Diego, CA, submitted.

[6] M. Haseli and J. Cortés, "Data-driven approximation of Koopman-invariant subspaces with tunable accuracy," in *American Control Conference*, New Orleans, LA, July 2021, pp. 469–474.

[7] M. Haseli and J. Cortés, "Fast identification of Koopman-invariant subspaces: parallel symmetric subspace decomposition," in *American Control Conference*, Denver, CO, July 2020, pp. 4545–4550.

[8] M. Haseli and J. Cortés, "Efficient identification of linear evolutions in nonlinear vector fields: Koopman invariant subspaces," in *IEEE Conf. on Decision and Control*, Nice, France, Dec. 2019, pp. 1746–1751.

[9] M. Haseli and J. Cortés, "Approximating the Koopman operator using noisy data: noise-resilient extended dynamic mode decomposition," in *American Control Conference*, Philadelphia, PA, July 2019, pp. 5499–5504.

ABSTRACT OF THE DISSERTATION

Data-Driven System Analysis Using the Koopman Operator: Eigenfunctions, Invariant Subspaces, and Accuracy Bounds

by

Masih Haseli

Doctor of Philosophy in Engineering Sciences (Mechanical Engineering)

University of California San Diego, 2022

Professor Jorge Cortés, Chair

Ranging from natural phenomena such as biological and chemical systems to artificial technologies such as mechanical and electronic devices, dynamical systems form an inseparable part of the world surrounding us. Understanding, modeling, predicting, and controlling such systems have always been the leading goals of science and engineering. While in the past centuries, the most advances in the field of dynamical systems were mainly analytical and based on limited observations, in the last decade, we have witnessed a rapid growth in our ability to gather, store, and process data. This data-driven revolution has imposed a high demand for new viewpoints and systematic structures that can effectively utilize the available modern tools. The Koopman operator theory for dynamical systems has recently gained widespread attention. Unlike the traditional state space methods, which explain the evolution of the states according to the dynamics, the Koopman operator characterizes the effect of the dynamics on functions in a linear function space. Despite the system being linear or nonlinear, its associated Koopman operator is always linear. This linearity proves to be extremely useful for algorithmic computations.

This dissertation is focused on the data-driven analysis of dynamical systems based on their associated Koopman operator. Given the infinite-dimensional nature of the Koopman operator, we aim to find finite-dimensional spaces on which the operator's action can be captured accurately. Such finite-dimensional spaces must be close to being invariant under the action of the Koopman operator; otherwise, the approximation will be erroneous. We provide data-driven algebraic methods to provably find the exact maximal Koopman-invariant subspace and all Koopman eigenfunctions in any arbitrary finite-dimensional space of functions. We also offer equivalent algorithms tailored for working with large and streaming data sets, as well as parallel computing hardware. Keeping in mind that an exact finite-dimensional linear model might not capture complete information for some systems, we also consider approximating subspaces to capture more information about the dynamics. We provide data-driven measures to quantify how close a linear space of functions is to being invariant under the Koopman operator. In addition, we provide an algebraic procedure that can approximate Koopman-invariant subspaces with tunable accuracy.

Chapter 1

Introduction

Our world is changing. We just need to think about the technologies that we trivially use nowadays. Many of them did not exist a decade ago, and some were even unimaginable. This rapid development is mainly fueled by advances in computing, sensing, data processing, data storage, cloud services, and algorithms. With these advances also come new challenges.

Almost all physical devices and computer programs ruling our world are dynamical systems. Thus understanding, designing, and controlling them are of utmost importance. However, the traditional methods to study dynamical systems are not designed to deal with large data sets, nor do they consider digital computers' limitations in dealing with creative geometric ideas. As a result, to effectively utilize the new capabilities at our disposal, we need new viewpoints of dynamical systems compatible with large data sets and adaptable to the shortcomings of digital computers in creative thinking.

Statistical and probabilistic approaches provide an alternative description for dynamical systems conducive to dealing with data and even incomplete information about the underlying dynamics. Even though such methods can extract useful information from data, they generally require some a priori knowledge about the system and the sampling schemes. Moreover, under such approaches, deriving mathematical guarantees for general nonlinear systems may be difficult if not impossible

Using neural networks to deal with data is another interesting approach that has gained widespread attention recently. Given sufficient data, neural networks can describe nonlinear dynamics with high accuracy, a feature that has led to their striking popularity over the past decade. Despite significant advances in neural network training algorithms, they are still considered computationally intensive. Perhaps the main drawback of neural networks is their complexity. The neural network models are generally highly nonlinear and difficult to analyze. As a result, to use neural networks in system analysis with theoretical guarantees, the underlying methods must be tailored to the dynamics, which is generally challenging and might require some basic assumptions or knowledge.

These reasons have motivated researchers to seek alternative strategies to capture the dynamics using data with minimum a priori information in a computationally efficient way that result in simple yet accurate models. Approximating the Koopman operator associated with a dynamical system is one of such strategies. The Koopman operator is a linear but generally infinite-dimensional operator that fully describes the behavior of the underlying dynamical system. Even though the linearity of the Koopman operator makes its spectral properties a powerful tool for analysis, its infinite-dimensional nature prevents the use of conventional linear algebraic tools developed to work with digital computers. One way to circumvent this issue is to identify finite-dimensional subspaces that are invariant under the

Koopman operator. This dissertation develops efficient data-driven methods to identify and approximate such subspaces accompanied by accuracy bounds and theoretical guarantees.

1.1 Literature Review

The Koopman operator [Koo31, KN32] is a linear but generally infinite-dimensional operator that provides an alternative view of dynamical systems by describing the effect of the dynamics on a function space. Being a linear operator enables one to use its spectral properties to capture and predict the behavior of nonlinear dynamical systems [Mez05, RMB$^+$09, BMM12]. Even though the Koopman operator is generally infinite-dimensional, its eigenfunctions still can be useful in finite-dimensional spaces. Given that the Koopman eigenfunctions evolve linearly on the system's trajectories, they can encode valuable information about the dynamics and the features of the vector field. The work in [MMM13] uses Koopman eigenfunctions to define natural action-angle coordinates for stable fixed points and shows that such coordinates are connected to specific types of Lyapunov functions. The work in [LM13] shows that certain Koopman eigenfunctions can be used as change of coordinates to put nonlinear systems with stable equilibria or periodic orbits in linear form on the basin of attraction. Interestingly, this linearization is simply a change of coordinates and is not obtained by embedding in infinite-dimensional space. Moreover, [MM16] studies the connection of Koopman operator and global stability of hyperbolic fixed points and periodic orbits, providing equivalence between global stability of attractors and the existence of certain Koopman eigenfunctions. As a practical and interesting byproduct, the analysis provides a recipe for construction of Lyapunov functions based on the aforementioned Koopman eigenfunctions.

The deep connection of the Koopman eigenfunctions with the system's behavior, and the linearity of the operator have lead to a wide variety of applications. [SWMB17,NM18] use the Koopman operator approaches for state estimation. [MG16,MG19,BRV19,KGB+17] rely on the linearity of the Koopman operator for system identification. The work [SVR16] uses the Koopman operator framework to study the sensor and actuator placement problem. Perhaps one of the most important applications of the Koopman paradigm is model reduction, see e.g. [Mez05], since the Koopman eigenfunctions naturally decompose the dynamic behavior. Such decomposition and model reduction can be calculated directly based on data acquired form the system [KNK+18, AK17, KNP+20].

Even though the Koopman operator is not defined for systems with input, it is still useful for control of dynamical systems. In [KKB21], the authors present a data-driven method for identification of Koopman eigenfunctions with applications in control systems. [KM18a] provides a class of approximate linear predictors based on Koopman operator and provides a control method using model predictive control (MPC) schemes. The works [HMV18, GP21] propose Koopman-based bilinear models for certain classes of control affine systems and provide appropriate methods for their stabilization. [SE17] studies the problem of control under uncertainty for a certain class of monotone systems and [LWJ21] proposes a data-based Koopman predictive control considering the system's uncertainty. [PK19] provides a model reduction and control scheme for partial differential equation (PDE) systems based on linear switched modeling. Interestingly, there have been some effort to connect the Koopman operator theory to well-known and widely used control concepts such as Lyapunov and barrier functions. The work in [ZB22b] provides a neural network based method to create control Lyapunov functions in Koopman paradigm and [FCAB20] provides a method to synthesize control barrier functions to ensure the safety of controlled systems. The rapid advances in Koopman-based control schemes have also found their way into robotic applications recently [ZB22a, AdlTM17, MCTM21, SK21].

Due to the infinite-dimensional nature of the Koopman operator, the digital implementation of the aforementioned applications is not possible unless one can find a way to represent the effect of the operator on finite-dimensional subspaces. The literature has explored several data-driven methods to find such finite-dimensional approximations, which can be divided into two main categories: projection-based methods and invariant-subspace methods. We discuss these methods next.

1.1.1 Projection-based Learning Methods

Projection methods fit a linear model to the data acquired from the system. The most popular approach in this category is Dynamic Mode Decomposition (DMD), first proposed to capture dynamical information from fluid flows [Sch10]. DMD uses linear algebraic methods to form a linear model from time series data and extract the dominant features of the flow. The work [CTR12] explores the properties of DMD and its connection with the Koopman operator, and [TRL$^+$14] generalizes it to work with non-sequential data snapshots.

As a result of the popularity of DMD, there have been several extensions for it in the literature. To deal with large data sets, [HWR14] provides a streaming method that iteratively updates its solution as new data become available and [ZRDC19] proposes a similar online approach for time-varying systems. Moreover, [AM19] proposes a variant of DMD that is compatible with parallel processing schemes as well as large and streaming data sets. The DMD's algebraic mechanism does not consider the existence of measurement noise in the data. To tackle this issue, the work in [DHWR16] studies the effect of noise on DMD and provides several indirect and direct methods to deal with it. In addition, [HRDC17] proposes a two-step algorithm, similar to total least squares method, to deal with measurement noise in DMD. Other extensions of DMD use sparsity promoting techniques [JSN14] and consider time-lagged data snapshots [CV17] to enhance the captured information. Given that DMD is generally designed to extract *dominant* features of the dynamics, it should be used with care for prediction purposes. The work in [LT20] analyzes the accuracy of long-term prediction by DMD and its variants.

Extended Dynamic Mode Decomposition (EDMD) [WKR15] is an important variations of DMD that lifts the states of the system to a (generally higher-dimensional) function space using a predefined dictionary of functions and finds the projection of the Koopman operator on that space. The work [KM18b] studies the convergence properties of EDMD to the Koopman operator as the number of data snapshots and dictionary elements go to infinity. Similarly to DMD, EDMD is specifically designed to work with exact data and experiments and simulations show that it may not work well with data corrupted with measurement noise. The work in [HC19a] presents a noise-resilient extension of EDMD based on element-wise weighted total least squares methods. The quality of EDMD directly depends on its dictionary. As a result, the dictionary must be chosen based on the dynamics. The work [MCTM21] provides a Taylor expansion method to enrich the dictionary for EDMD to achieve lower errors in long-term predictions. [ZZ21] uses finite element methods to systematically learn Koopman approximations on finite-dimensional spaces. This work also provides a variant of EDMD to learn the Koopman generator combined with finite element methods. Another area worth noting regarding EDMD is characterizing its accuracy while finitely many data samples are available. [NPP$^+$21] addresses this important issue by providing several probabilistic finite sample bounds for approximation of the Koopman operator both for deterministic and stochastic systems.

The aforementioned projection methods provide linear higher-dimensional approximations for the underlying dynamics that are, however, not suitable for long-term predictions, since they are generally not exact. This issue can be tackled by finding subspaces that are invariant under the Koopman operator, since the acquired linear models are exact over them. This is the subject of the second group of approaches.

1.1.2 Invariant Subspace Methods

On a finite-dimensional Koopman-invariant subspace, the action of the Koopman operator can be captured by a matrix product. Hence, finding such invariant subspaces has major theoretical and practical implications. The works [BBPK16, JY18] use the system's model to analytically derive Koopman-invariant subspaces for specific classes of nonlinear systems. For the data-driven setting, [LDBK17, TKY17, YKH19, LKB18] provide approaches to find functions that span Koopman-invariant subspaces using neural networks. A alternative approach is to find Koopman eigenfunctions which span Koopman-invariant subspaces. The work [KKB21] provides a data-driven method to identify Koopman eigenfunctions.

Moreover, [KM20] provides theoretical results and empirical methods based on multi-step predictions on nonrecurrent sets to find Koopman eigenfunctions. A different approach to approximate Koopman invariant subspaces is to use data combined with other information about the system. [BBK$^+$21, FYR$^+$21] approximate finite-dimensional Koopman models relying on knowledge about the system's attractors and their stability.

1.2 Statement of Contributions

This dissertation studies the data-driven identification of nonlinear systems through their associated Koopman operator. We study the algebraic properties of the Koopman eigenfunctions and their relation to data acquired from the system's trajectories. Relying on our findings, we provide algebraic algorithms to identify Koopman eigenfunctions and invariant subspaces with accuracy and convergence guarantees. Moreover, we provide specific methods to tackle practical challenges such as dealing with large and streaming data sets as well as using parallel computation to speed up the identification process. Next, we briefly review the contributions made in each chapter.

Chapter 3: We present data-driven methods to identify Koopman eigenfunctions and Koopman-invariant subspaces associated with a potentially nonlinear dynamical system. First, we study the properties of the standard Extended Dynamic Mode Decomposition (EDMD) method regarding the identification of Koopman eigenfunctions. We prove that EDMD correctly identifies all the Koopman eigenfunctions in the span of the predefined dictionary. This necessary condition however is not sufficient, i.e., the functions identified by the EDMD method are not necessarily Koopman eigenfunctions. This motivates our next contribution, which is a necessary and sufficient condition that characterizes the functions that evolve linearly according to the available data snapshots. This condition is based on the application of EDMD forward and backward in time. The identified functions are not necessarily Koopman eigenfunctions, since one can only guarantee that they evolve linearly on the available data (but not necessarily starting anywhere in the state space). However, we prove that under reasonable assumptions on the density of the sampling, the identified functions are Koopman eigenfunctions almost surely.

Our next contribution seeks to provide computationally efficient ways of identifying Koopman eigenfunctions and Koopman-invariant subspaces. In fact, checking the aforementioned necessary and sufficient condition requires one to calculate and compare the eigendecomposition of two potentially large matrices, which can be computationally cumbersome. Moreover, even though the subspace spanned by all the eigenfunctions in the span of the original dictionary is Koopman-invariant, it might not be maximal. To address these limitations, we propose the Symmetric Subspace Decomposition (SSD) strategy, which is an iterative method to find the maximal subspace that remains invariant under the application of dynamics (and its associated Koopman operator) according to the available data. We prove that SSD also finds all the functions that evolve linearly in time according to the available data. Moreover, we prove that under the same conditions on the sampling density, the SSD strategy identifies the maximal Koopman-invariant subspace in the span of the original dictionary almost surely.

Our next contribution is motivated by applications where the data becomes available in an online fashion. In such scenarios, at any given time step, one would need to perform SSD on all the available data received up to that time. Performing SSD requires the calculation of several singular value decompositions for matrices that scale with the size of the data, in turn requiring significant memory capabilities. To address these shortcomings, we propose the Streaming Symmetric Subspace Decomposition (SSSD) strategy, which refines the calculated Koopman-invariant subspaces each time it receives new data and deals with matrices of fixed and relatively small size (independent of the size of the data). We prove that SSSD and SSD methods are equivalent, in the sense that for a given dataset, they both identify the same maximal Koopman-invariant subspace.

Our last contribution is motivated by the fact that, in some cases the predefined dictionary does not contain sufficient eigenfunctions to capture important information from the dynamics. To address this, we provide an extension of SSD, termed Approximated-SSD, enabling us to approximate Koopman eigenfunctions and invariant subspaces, and show how its accuracy can be tuned using a design parameter.

Chapter 4: We present a counterpart to the Symmetric Subspace Decomposition (SSD) algorithm presented in Chapter 3. The proposed method is compatible with parallel processing hardware. Our starting point is a group of processors communicating through a directed graph, with each processor aware of a common dictionary of functions and equipped with a local set of data snapshots acquired from the dynamics. We introduce the Parallel Symmetric Subspace Decomposition (P-SSD) algorithm to find the maximal Koopman-invariant subspace and all the Koopman eigenfunctions in the finite-dimensional linear functional space spanned by the dictionary. The proposed strategy has each processor refine its estimate of the invariant dictionary by iteratively employing the information received from its neighbors to prune it. We show that the P-SSD algorithm reaches an equilibrium in a finite number of time steps for any (possibly time-varying) network topology and carefully characterize the properties of the agents' dictionary iterates along its execution, particularly in what concerns monotonicity of the associated subspaces. We also establish that the globally reachable processors in the communication digraph find the same solution as the SSD algorithm would if all data was centrally available at a single processor. This allows us to conclude that the P-SSD algorithm finds the maximal Koopman-invariant subspace in the space spanned by the original dictionary if the network topology is strongly connected. Finally, we conclude by characterizing the algorithm's time, computational, and communication complexity, demonstrating its computational advantage over SSD, and showing its robustness against communication failures and packet drops.

Chapter 5: In Chapters 3-4, we proposed algorithms to identify the maximal Koopman-invariant subspace of any arbitrary finite-dimensional space of functions. The Koopman-based models on such invariant subspaces are exact. However, in general, capturing complete information about a nonlinear system's behavior might not be possible on finite-dimensional Koopman-invariant subspaces. A practical remedy for this issue is to *approximate* models in order to capture more (although inexact) information at the cost of introducing some error in the models. This is especially relevant since most practical applications only need reasonably accurate (but possibly inexact) models. Our goal in Chapter 5 is to provide methods that can identify Koopman-based models with tunable level of accuracy and expressiveness.

Our main result, consists of the synthesis of a computational procedure, termed Tunable Symmetric Subspace Decomposition (T-SSD), that given an *arbitrary* finite-dimensional functional space, balances the trade-off between the expressiveness of its subspaces and the accuracy of the Koopman approximations on them.

The roadmap of supporting contributions leading to the design and full characterization of T-SSD is as follows. Our first contribution builds on the observation that the proximity of a functional space to being invariant is a measure of its (and consequently its members') prediction accuracy under finite-dimensional Koopman approximations, as an exact invariant subspace leads to exact predictions of the evolution of observables. We introduce the novel notion of ϵ-apart spaces to measure invariance proximity using data snapshots sampled from the trajectories of the unknown dynamics. Using this notion, and given an arbitrary finite-dimensional functional space spanned by a dictionary of functions, we formulate our objective as that of finding a parametric family of subspaces whose value of the parameter determines the desired level of invariance proximity. This parametric family can be viewed as balancing invariance proximity (i.e., prediction accuracy) and the dimension of the subspace (i.e., expressiveness).

Given a desired accuracy parameter, our second contribution is the design of T-SSD as an algorithmic procedure that finds a functional space satisfying the desired accuracy by iteratively removing the functions in the span of the original dictionary that violate the desired accuracy. We show that T-SSD terminates in finite iterations and characterize its computational complexity. Moreover, we show that its identified subspaces contain the maximal Koopman-invariant subspace and all Koopman eigenfunctions in the span of the original dictionary. We also show that the accuracy parameter bounds the relative root mean square prediction error for all (uncountably many) functions in the identified subspace. This advantage of the T-SSD algorithm in deriving accuracy bounds on the prediction of individual functions independently of linear changes of coordinates stems from focusing on the subspaces instead of their basis. Our next contribution establishes that both Extended Dynamic Mode Decomposition and Symmetric Subspace Decomposition algorithms are particular cases of T-SSD.

Our final contribution is a computationally efficient version of T-SSD with drastically lower computational complexity when the number of data snapshots is significantly larger than the dimension of the original dictionary.

Chapter 6: In the previous chapters, we proposed algebraic algorithms to find finite-dimensional subspaces that are close to being invariant under the action of the Koopman operator. The Koopman-based model then is calculated by applying the Extended Dynamic Mode Decomposition (EDMD) algorithm on the identified subspaces by our methods. Many existing methods in the literature, however, use optimization or neural-network based methods to find the correct dictionary and subspace for EDMD. In Chapter 6, we aim to find an appropriate accuracy measure for EDMD that can be used as a cost function in optimization and neural-network based methods.

Our starting point is the observation that the residual error of EDMD, typically used for dictionary learning, does not necessarily measure the quality of the subspace spanned by the dictionary and, consequently, the prediction accuracy of EDMD on the subspace. To illustrate this point, we provide an example showing that one can choose a sequence of dictionaries spanning the same subspace that make the residual error arbitrarily close to zero. This motivates our goal of identifying better measures to assess the EDMD's prediction accuracy and its dictionary's quality. We define the notion of the consistency matrix and its spectral radius, which we term consistency index, which measures the deviation of the EDMD solutions forward and backward in time from being the inverse of each other. This is justified by the fact that if a subspace is Koopman invariant, the EDMD solutions applied forward and backward in time are the inverse of each other. We characterize various algebraic properties of the consistency index and show that it only depends on the data and the space spanned by the dictionary, and is hence invariant under changes of basis. We also establish that the square root of the consistency index in fact provides a tight upper bound on the relative root mean square EDMD prediction error of all functions in the span of the dictionary.

Chapter 2

Preliminaries

In this chapter, we introduce the notations used in the dissertation. Moreover, we review basic concepts from graph theory, Koopman operator theory, as well as the Extended Dynamic Mode Decomposition method.

2.1 Notations

Here, we present the notations used throughout this dissertation.

2.1.1 Sets and Functions

Given a set S, we denote its complement by S^c. Given sets S_1 and S_2, $S_1 \subseteq S_2$ and $S_1 \subsetneq S_2$ respectively mean that S_1 is a subset and proper subset of S_2. We denote by $S_1 \cap S_2$ and $S_1 \cup S_2$, the intersection and union of S_1 and S_2. Moreover, we define $S_1 \setminus S_2 := S_1 \cap S_2^c$. Given a sequence of sets $\{S_i\}_{i=1}^{\infty}$, we denote its superior and inferior limits by $\limsup_{i\to\infty} S_i$ and $\liminf_{i\to\infty} S_i$, respectively. Given functions f and g with appropriate domains and co-domains, $f \circ g$ denotes their composition. We also refer to the set consisting of all continuous strictly increasing functions $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $\alpha(0) = 0$ as class-$\mathcal{K}$.

2.1.2 Number Systems, Vectors, and Matrices

We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{C}$, the sets of natural, nonnegative integer, real, nonnegative real, and complex numbers respectively. We use j to denote the imaginary unit (the solution of $x^2 + 1 = 0$). Given integers a, b we use $a \bmod b$ to represent the remainder of division of a by b.

For a matrix $A \in \mathbb{C}^{m \times n}$, we denote the sets comprised of its rows by $\text{rows}(A)$, its columns by $\text{cols}(A)$, the number of its rows by $\sharp\text{rows}(A)$, and the number of its columns by $\sharp\text{cols}(A)$, respectively. In addition, we denote its pseudo-inverse, transpose, complex conjugate, conjugate transpose, Frobenius norm, and range space by $A^\dagger$, A^T, $\bar{A}$, A^H, $\|A\|_F$, and $\mathcal{R}(A)$, respectively. Moreover, for $1 \leq i < k \leq m$, we denote by $A_{i:k}$ the matrix formed with the ith to kth rows of A. Also, $A_{i,j}$ denotes the ijth element of A. Given matrices $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{m \times d}$, we denote by $[A, B] \in \mathbb{C}^{m \times (n+d)}$ the matrix created by concatenating A and B.

Given the square matrix B, we use B^{-1} to denote its inverse. We denote by $\text{spec}(B)$ the set comprised of the eigenvalues of B. Similarly, $\text{spec}_{\neq 0}(B)$ denotes the set of nonzero eigenvalues of A. Moreover, $\text{sprad}(B) := \max\{|\lambda| \mid \lambda \in \text{spec}(B)\}$ is the spectral radius of B. If $\text{spec}(B) \subset \mathbb{R}$, then $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$ denote the smallest and largest eigenvalues of B. We denote by $\mathbf{0}_{m \times n}$ and I_n, the $m \times n$ zero matrix and the identity matrix of size n respectively (we omit the indices when the context is clear).

For $v \in \mathbb{C}^n$, we denote its real and imaginary parts by $\mathrm{Re}(v)$ and $\mathrm{Im}(v)$, and its 2-norm as $\|v\|_2 := \sqrt{v^H v}$. The angle between vectors $v, w \in \mathbb{R}^n$ is denoted by $\angle(v, w)$.

2.1.3 Vector Spaces

Given the vector space $\mathcal{V}$ defined on the field ($\mathbb{C}$ or $\mathbb{R}$) [1], $\dim \mathcal{V}$ denotes its dimension. Moreover, given a set $\mathcal{S} \subseteq \mathcal{V}$, $\mathrm{span}(\mathcal{S})$ is a vector space comprised of all linear combinations of elements in $\mathcal{S}$. If vectors $v, w \in \mathbb{R}^n$ and vector spaces $\mathcal{V}, \mathcal{W} \subseteq \mathbb{R}^n$ are orthogonal, we write $v \perp w$ and $\mathcal{V} \perp \mathcal{W}$. Moreover, $\mathcal{V}^\perp$ denotes the orthogonal complement of $\mathcal{V}$. For a vector space $\mathcal{V} \subseteq \mathbb{R}^m$, $\mathcal{P}_\mathcal{V}$ denotes the orthogonal projection operator on $\mathcal{V}$. For convenience, we denote the orthogonal projection operator on the range space of a matrix A by $\mathcal{P}_A$, which takes the form $\mathcal{P}_A w = AA^\dagger w$, for $w \in \mathbb{R}^m$. We define the sum of vector spaces $\mathcal{V}_1, \mathcal{V}_2$ by $\mathcal{V}_1 + \mathcal{V}_2 := \{v_1 + v_2 | v_1 \in \mathcal{V}_1 \wedge v_2 \in \mathcal{V}_2\}$.

2.2 Graph Theory

Our exposition here mainly follows [BCM09, LMNB17]. Given a set V comprised of m *nodes* and a set $E \subseteq V \times V$ comprised of ordered pairs of nodes called *edges*, the pair $G = (V, E)$ defines a *directed graph (digraph)* on nodes V and edges E. We say node i is an *in-neighbor* of node j and node j is an *out-neighbor* of node i if $(i, j) \in E$. We denote by $\mathcal{N}_{\mathrm{in}}(i)$ and $\mathcal{N}_{\mathrm{out}}(i)$ the sets of in- and out-neighbors of i, respectively. A *directed path* with *length* l is an ordered sequence of $l + 1$ nodes such that the ordered pair of every two

[1] All vector spaces in this dissertation are defined on the field of complex numbers or real numbers. We omit mentioning the field where the context is clear.

consecutive nodes is an edge of the digraph. A path is *closed* if its first and last nodes are the same. A node is called *globally reachable* if there is a directed path from every other node to it. A digraph is *strongly connected* if all of its nodes are globally reachable. Given two nodes i, j in a digraph, the *distance* from i to j, denoted $\operatorname{dist}(i, j)$, is the length of the shortest directed path from i to j. If there is no such directed path, the distance is ∞. Given two digraphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, we define their *composition* as $G_2 \circ G_1 = (V, E_\circ)$ such that $E_\circ = \{(i, j) \mid \exists k \in V \text{ with } (i, k) \in E_1 \wedge (k, j) \in E_2\}$. A finite sequence of digraphs $\{G_i = (V, E_i)\}_{i=1}^{k}$ is *jointly strongly connected* if $G_k \circ \cdots \circ G_1$ is strongly connected. An infinite sequence of digraphs (or a time-varying digraph) $\{G_i = (V, E_i)\}_{i=1}^{\infty}$ is *repeatedly jointly strongly connected* if there exists $l, \tau \in \mathbb{N}$ such that for every $k \in \mathbb{N}_0$, the sequence of digraphs $\{G_{\tau+kl+j}\}_{j=0}^{l-1}$ is jointly strongly connected.

2.3 Koopman Operator

Here, we introduce the Koopman operator associated with discrete- and continuous-time systems and its properties. Our exposition mainly follows [BMM12].

2.3.1 Discrete-Time Systems

Consider a nonlinear, time-invariant map $T : \mathcal{M} \to \mathcal{M}$ on $\mathcal{M} \subseteq \mathbb{R}^n$, defining the dynamical system

$$x^+ = T(x). \tag{2.1}$$

The dynamics (2.1) acts on the points in the state space $\mathcal{M}$ and generates trajectories of the system. The Koopman operator, on the other hand, provides an alternative approach to analyze (2.1) based on evolution of functions (also known as *observables*) defined on $\mathcal{M}$ and taking values in $\mathbb{C}$. Formally, let $\mathcal{F}$ be a linear space of functions (defined on the field $\mathbb{C}$) from $\mathcal{M}$ to $\mathbb{C}$ which is closed under composition with T, i.e.,

$$f \circ T \in \mathcal{F}, \quad \forall f \in \mathcal{F}. \tag{2.2}$$

The Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ associated with (2.1) is

$$\mathcal{K} f = f \circ T. \tag{2.3}$$

A closer look at the definition of the Koopman operator shows that it advances the observables in time, i.e.,

$$\mathcal{K} f(x) = f \circ T(x) = f(x^+), \quad \forall x \in \mathcal{M}. \tag{2.4}$$

This equation shows how the Koopman operator encodes the dynamics on the functional space $\mathcal{F}$. The operator is linear as a direct consequence of linearity in $\mathcal{F}$, i.e., for every $f_1, f_2 \in \mathcal{F}$ and $c_1, c_2 \in \mathbb{C}$,

$$\mathcal{K}(c_1 f_1 + c_2 f_2) = c_1 \mathcal{K}(f_1) + c_2 \mathcal{K}(f_2). \tag{2.5}$$

Assuming $\mathcal{F}$ contains the functions describing the states of the system, $g_i(x) = x_i$ with $i \in \{1, \ldots, n\}$, the Koopman operator fully characterizes the global features of the dynamics in a linear fashion. Moreover, the operator might be (and generally is) infinite dimensional either by choice of $\mathcal{F}$ or due to closedness requirement in (2.2).

Being linear, one can naturally define its eigendecomposition. A function $\phi \in \mathcal{F}$ is an *eigenfunction* of $\mathcal{K}$ associated with *eigenvalue* $\lambda \in \mathbb{C}$ if

$$\mathcal{K}\phi = \lambda\phi. \tag{2.6}$$

The combination of (2.4) and (2.6) leads to a significant property of the Koopman operator: the linear evolution of its eigenfunctions in time. Formally, given an eigenfunction ϕ with eigenvalue λ,

$$\phi(x^+) = \phi \circ T(x) = \mathcal{K}\phi(x) = \lambda\phi(x). \tag{2.7}$$

The linear evolution of eigenfunctions (cf. (2.7)), together with the linearity of the operator in (2.5), enables us to use spectral properties to analyze the nonlinear system (2.1). Given a set of eigenpairs $\{(\lambda_i, \phi_i)\}_{i=1}^{N_k}$ such that $\mathcal{K}\phi_i = \lambda_i\phi_i, i \in \{1, \ldots, N_k\}$, one can describe the evolution of every function f in $\operatorname{span}(\{\phi_i\}_{i=1}^{N_k})$, i.e., $f = \sum_{i=1}^{N_k} c_i\phi_i$, for some $\{c_i\}_{i=1}^{N_k} \subset \mathbb{C}$, as

$$f(x(k)) = \sum_{i=1}^{N_k} c_i\lambda_i^k\, \phi_i(x(0)), \quad \forall k \in \mathbb{N}_0, \tag{2.8}$$

where $\{x(l)\}_{l=0}^{\infty}$ is the trajectory of the system starting from the initial condition $x(0)$. The constants $\{c_i\}_{i=1}^{N_k}$ are called *Koopman modes*. It is important to note that one might need to use infinitely many eigenfunctions to fully describe the behavior of the dynamical system.

Another important notion in the analysis of the Koopman operator is the invariance of subspaces under its application. Formally, a subspace $\mathcal{S} \subseteq \mathcal{F}$ is *Koopman-invariant* if for every $f \in \mathcal{S}$ we have $\mathcal{K}f \in \mathcal{S}$. Furthermore, $\mathcal{S}$ is *maximal Koopman-invariant* in $\mathcal{L} \subseteq \mathcal{F}$ if it contains every Koopman-invariant subspace in $\mathcal{L}$. Naturally, a set comprised of Koopman eigenfunctions spans a Koopman-invariant subspace.

2.3.2 Continuous-Time Systems

Consider the following continuous-time dynamical system [2] defined on the state space $\mathcal{M} \in \mathbb{R}^n$

$$\dot{x} = g(x). \tag{2.9}$$

Also, let $\mathcal{G}^t(x_0)$ be the flow map associated with the dynamics mapping the initial condition x_0 to $\mathcal{G}^t(x_0)$, the solution of the system at time t. Similarly to the the discrete-time case (cf. (2.3)), for each time $t \in \mathbb{R}_{\geq 0}$, the flow map induces a Koopman operator

$$\mathcal{K}^t f = f \circ \mathcal{G}^t, \;\; \forall f \in \mathcal{F}.$$

The set $\{\mathcal{K}^t\}_{t \in \mathbb{R}_{\geq 0}}$ is known as the *Koopman semigroup* and its infinitesimal generator is often used to describe the behavior of the system.

[2]We assume the dynamics has a unique solution starting from every point in the state space.

Remark 2.3.1. ***(In this dissertation we consider discrete-time or sampled continuous-time systems).*** In this dissertation, we generally consider discrete-time systems in the form of (2.1). However, the material presented here can be used for continuous-time systems. In fact, if the trajectories of system (2.9) are sampled with the sampling time Δt, the presented data-driven algorithms in this dissertation capture the behavior of the operator $\mathcal{K}^{\Delta t}$ associated with the flow map $\mathcal{G}^{\Delta t}$. $\square$

2.4 Extended Dynamic Mode Decomposition

Here, we briefly introduce the Extended Dynamic Mode Decomposition (EDMD) method following [WKR15] [3]. As mentioned earlier, the infinite-dimensional property of the Koopman operator prevents its direct use in practical data-driven settings. This leads naturally to constructing finite-dimensional approximations. EDMD is a method for constructing finite-dimensional Koopman approximations from data. EDMD uses a dictionary $D : \mathcal{M} \to \mathbb{R}^{1\times N_d}$ containing N_d functions in the form of

$$D(x) = [d_1(x), \dots, d_{N_d}(x)].$$

To capture the behavior of the dynamics (and hence the Koopman operator), EDMD uses data matrices $X, Y \in \mathbb{R}^{N\times n}$ containing N data snapshots gathered from the trajectories of

[3]For convenience, the notation employed in this dissertation is slightly different from the traditional notation in the literature.

the system (2.1),

$$y_i = T(x_i), \forall i \in \{1, \dots, N\}, \tag{2.10}$$

where x_i^T and y_i^T correspond to the ith rows of X and Y. EDMD approximates the action of the Koopman operator on span(D) by solving the least-squares problem

$$\underset{K}{\text{minimize}} \|D(Y) - D(X)K\|_F \tag{2.11}$$

which has the closed-form solution

$$K_{\text{EDMD}} = \text{EDMD}(D, X, Y) := D(X)^{\dagger} D(Y). \tag{2.12}$$

Throughout this dissertation, we rely on the following basic assumption.

Assumption 2.4.1. ***(Full Rank Dictionary Matrices).*** *$D(X)$ and $D(Y)$ have full column rank.* □

Assumption 2.4.1 implies that the functions in D are linearly independent, i.e., they form a basis for span(D) and the data are diverse enough to distinguish between the elements of D. Moreover, Assumption 2.4.1 ensures that K_{EDMD} is the unique solution for (2.11).

One can use the eigendecomposition of K_{EDMD} to approximate Koopman eigenfunctions. Formally, given the eigenvector $v \in \mathbb{C}^{N_d} \setminus \{0\}$ of K_{EDMD} with eigenvalue $\lambda \in \mathbb{C}$, we define the following approximated Koopman eigenfunction with eigenvalue λ as

$$\phi(\cdot) = D(\cdot)v. \tag{2.13}$$

Moreover, one can use K_{EDMD} to approximate the action of the operator on $\text{span}(D)$. Given a function $f \in \text{span}(D)$ in the form of $f(\cdot) = D(\cdot)v_f$ for $v_f \in \mathbb{C}^{N_d}$, we define the predictor function for $\mathcal{K}f$ as

$$\mathfrak{P}_{\mathcal{K}f}(\cdot) = D(\cdot)K_{\text{EDMD}}v_f. \tag{2.14}$$

It should be noted that, the predictor (2.14) can be viewed as L_2-orthogonal projection of $\mathcal{K}f$ on $\text{span}(D)$ calculated using an empirical measure defined based on the rows of the data matrix X (see e.g. [KM18b, Theorem 1]). As a result of this projection, the quality of this predictor directly depends on the choice of the dictionary. If $\text{span}(D)$ is Koopman-invariant, i.e., if $\mathcal{K}g \in \text{span}(D)$ for all $g \in \text{span}(D)$, then the EDMD predictor (2.14) is exact. Otherwise, the prediction is inexact for some functions in the space. In this dissertation, we aim to find suitable dictionaries which lead to accurate EDMD prediction.

Chapter 3

Data-driven Identification of Koopman Eigenfunctions and Invariant Subspaces

This chapter develops data-driven methods to identify eigenfunctions of the Koopman operator associated to a dynamical system and subspaces that are invariant under the operator. Given any arbitrary finite-dimensional space of functions spanned by a dictionary (basis), we propose a necessary and sufficient condition to identify all Koopman eigenfunctions in the space based on the application of the Extended Dynamic Mode Decomposition (EDMD) forward and backward in time. Moreover, we propose the Symmetric Subspace Decomposition (SSD) algorithm, an iterative method which provably identifies the maximal Koopman-invariant subspace and the Koopman eigenfunctions in the span of the dictionary. We also introduce the Streaming Symmetric Subspace Decomposition (SSSD) algorithm, an online extension of SSD that only requires a small, fixed memory and incorporates new data as is received. Finally, we propose an extension of SSD that approximates Koopman eigenfunctions and invariant subspaces when the dictionary does not contain sufficient informative eigenfunctions.

3.1 Problem Statement

Consider a nonlinear, time-invariant, continuous map $T : \mathcal{M} \rightarrow \mathcal{M}$ on $\mathcal{M} \subseteq \mathbb{R}^n$, defining the dynamical system

$$x^+ = T(x). \tag{3.1}$$

Moreover, let $\mathcal{K}$ be its associated Koopman operator defined on the linear space of functions $\mathcal{F}$. Our goal is to find the maximal Koopman-invariant subspace and all Koopman eigenfunctions associated with system (3.1) in any arbitrary finite-dimensional subspace of $\mathcal{F}$ using data gathered from the system's trajectories. Formally, let $D : \mathcal{M} \rightarrow \mathbb{R}^{1\times N_d}$ be a dictionary of N_d functions in $\mathcal{F}$ such that $D(\cdot) = [d_1(\cdot), \dots, d_{N_d}(\cdot)]$ where $\{d_i\}_{i=1}^{N_d} \subset \mathcal{F}$. Moreover, let $X, Y \in \mathbb{R}^{N\times n}$ be matrices comprised of N data snapshots such that

$$y_i = T(x_i),\ i \in \{1, \dots, N\}, \tag{3.2}$$

where x_i^T and y_i^T are ith rows of X and Y, respectively. Our main goal is two-fold:

Problem 3.1.1. ***(Main Problem).*** *Given dictionary D and data matrices X and Y, we* *aim to*

(i) find all the Koopman eigenfunctions in span(D)*;*

(ii) find a basis for the maximal Koopman-invariant subspace in span(D)*.* □

Note that (i) and (ii) in Problem 3.1.1 are closely related. The eigenfunctions found by solving Problem 3.1.1(i) span Koopman-invariant subspaces. Those invariant subspaces however might not be maximal. This mild difference between Problem 3.1.1(i) and Problem 3.1.1(ii) requires the use of different solution approaches. Since we are dealing with finite-dimensional linear subspaces, we aim to use linear algebra instead of optimization-based methods, which are widely used for solving these types of problems. This enables us to directly use computationally efficient linear algebraic packages that optimization methods rely on.

We also pursue two secondary problems in order to improve the applicability of our methods in practical settings.

Problem 3.1.2. ***(Secondary Problems).*** *Extend the solutions for Problem 3.1.1 to the following settings:*

(i) updating the results sequentially when data is available in streaming format;

(ii) approximating Koopman eigenfunctions and invariant subspace when span(D) *does not contain informative exact eigenfunctions.* □

Throughout this chapter, we rely on the following standard assumption regarding the dictionary snapshots.

Assumption 3.1.3. ***(Full Column Rank Dictionary Matrices).*** *The matrices $D(X)$ and $D(Y)$ have full column rank.* □

Assumption 3.1.3 is reasonable: in order to hold, the dictionary functions must be linearly independent, i.e., the functions must form a basis for span(D). Moreover, the assumption requires the set of initial conditions rows(X) to be diverse enough to capture important characteristics of the dynamics. Our treatment here relies on the Extended Dynamic Mode Decomposition (EDMD) algorithm (cf. Section 2.4), which is not specifically designed to work with data corrupted with measurement noise. Hence, we assume access to data with high signal-to-noise ratio. In practice, one might need to pre-process the data to use the algorithms proposed here.

3.2 Identifying Koopman Eigenfunctions by Forward and Backward Extended Dynamic Mode Decomposition

Here, we investigate the capabilities and limitations of the EDMD method regarding the identification of Koopman eigenfunctions. Throughout this chapter, we use the following notations to represent the EDMD matrices applied on data matrices X and Y forward and backward in time

$$K_F = \text{EDMD}(D, X, Y), \quad K_B = \text{EDMD}(D, Y, X).$$

The next result shows that EDMD is not only able to capture Koopman eigenfunctions but also all the functions that evolve linearly according to the available data.

Lemma 3.2.1. ***(EDMD Captures the Koopman Eigenfunctions in the Span of the Dictionary).*** *Suppose Assumption 3.1.3 holds. Let $f(\cdot) = D(\cdot)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$.*

(i) Let f evolve linearly according to the available data, i.e., there exists $\lambda \in \mathbb{C}$ such that $f(y_i) = f \circ T(x_i) = \lambda f(x_i)$ for every $i \in \{1, \dots, \sharp\mathrm{rows}(X)\}$, where x_i and y_i are related through (3.2)*. Then, the vector v is an eigenvector of K_F with eigenvalue λ;*

(ii) Let f be an eigenfunction of the Koopman operator with eigenvalue λ. Then, the vector v is an eigenvector of K_F with eigenvalue λ.

Proof. (i) Based on the linear evolution of f, we have $D(Y)v = \lambda D(X)v$. Moreover, using the closed-form solution of EDMD, we have $K_F\, v = D(X)^\dagger D(Y)v = \lambda D(X)^\dagger D(X)v = \lambda v$, where the last equality follows from Assumption 3.1.3.

(ii) Based on the definition of Koopman eigenfunction, we have $f(x^+) = \lambda f(x)$. Since this linear evolution reflects in data snapshots, we have $f(y_i) = \lambda f(x_i)$ for every $i \in \{1, \dots, \sharp\mathrm{rows}(X)\}$ where x_i^T and y_i^T are the ith rows of X and Y respectively. The rest follows from (i). $\square$

Despite its simplicity, this result provides significant insight into the EDMD method. Lemma 3.2.1 shows that EDMD can capture eigenfunctions in the span of the dictionary even if the underlying subspace is not Koopman invariant. In the literature, it is well known that the (E)DMD method can capture physical constraints, conservation laws, and other properties of the underlying system, which actually correspond to Koopman eigenfunctions, e.g., see [WKR15, KNP+20]. We note that Lemma 3.2.1 is a generalization of [TRL+14, Theorem 1] to EDMD when the underlying system is not necessarily linear (or cannot be approximated by a linear system accurately) and the underlying subspace is not Koopman invariant. The next result shows that EDMD accurately predicts the evolution of functions in the span of Koopman eigenfunctions evaluated on the system's trajectories.

Proposition 3.2.2. ***(EDMD Accurately Predicts Evolution of any Linear Combination of Eigenfunctions on System's Trajectories).*** *Let $f(\cdot) = D(\cdot)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$. Assume f is in the span of eigenfunctions $\{\phi_i\}_{i=1}^m \subset \operatorname{span}(D)$ with corresponding eigenvalues $\{\lambda_i\}_{i=1}^m \subset \mathbb{C}$. Then, given any trajectory $\{x(j)\}_{j=0}^\infty$ of* (3.1),

$$f(x(j)) = D(x(0))K_F^j v, \ \forall j \in \mathbb{N}_0. \tag{3.3}$$

Proof. Since $f \in \operatorname{span}(\{\phi_i\}_{i=1}^m)$, there exist scalars $\{c_i\}_{i=1}^m \subset \mathbb{C}$ such that

$$f = \sum_{i=1}^m c_i \phi_i. \tag{3.4}$$

Since $\{\phi_i\}_{i=1}^m \subset \operatorname{span}(D)$, there exist vectors $\{w_i\}_{i=1}^m \subset \mathbb{C}^{N_d}$ such that

$$\phi_i(\cdot) = D(\cdot)w_i, \ \forall i \in \{1, \dots, m\}. \tag{3.5}$$

Combining (3.4) and (3.5) with the definition of f, we deduce that $\sum_{i=1}^{m} c_i w_i = v$. Now,

$$\begin{aligned} D(x(0))K_F^j v &= D(x(0))K_F^j \sum_{i=1}^{m} c_i w_i \\ &= D(x(0)) \sum_{i=1}^{m} c_i \lambda_i^j w_i, \end{aligned}$$

where in the last equality we have used Lemma 3.2.1(ii) for the eigenfunctions ϕ_is. Using (3.5),

$$D(x(0))K_F^j v = \sum_{i=1}^{m} c_i \lambda_i^j \phi_i(x(0)) = \sum_{i=1}^{m} c_i \phi_i(x(j)),$$

where in the second equality we have used the linear temporal evolution of Koopman eigenfunctions (2.7). The proof is now complete by noting that the previous equation holds for all $j \in \mathbb{N}_0$ and its the right-hand side is equal to $f(x(j))$ based on (3.4). $\square$

Lemma 3.2.1 provides a necessary condition for the identification of Koopman eigenfunctions. This condition however is not sufficient, see e.g. [HC19b, Example IV.3] for a counter example. Interestingly, if a function evolves linearly forward in time, it also evolves linearly backward in time. The next result shows that checking this observation provides a necessary and sufficient condition for identification of functions that evolve linearly in time according to the available data.

Theorem 3.2.3. ***(Identification of Linear Evolutions by Forward and Backward EDMD).*** *Suppose Assumption 3.1.3 holds. Let $f(\cdot) = D(\cdot)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$. Then $f \circ T(x_i) = f(y_i) = \lambda f(x_i)$ for some $\lambda \in \mathbb{C} \setminus \{0\}$ and for all $i \in \{1, \dots, \sharp\text{rows}(X)\}$ if and only* *if v is an eigenvector of K_F with eigenvalue λ, and an eigenvector of K_B with eigenvalue λ^{-1}.*

Proof. ($\Leftarrow$): Using the closed-form solutions of the EDMD problem and Assumption 3.1.3, one can write,

$$K_F = (D(X)^T D(X))^{-1} D(X)^T D(Y),$$
$$K_B = (D(Y)^T D(Y))^{-1} D(Y)^T D(X).$$

Using these along with the definition of the eigenpair,

$$\lambda D(X)^T D(X) v = D(X)^T D(Y) v, \tag{3.7a}$$
$$\lambda^{-1} D(Y)^T D(Y) v = D(Y)^T D(X) v. \tag{3.7b}$$

By multiplying (3.7a) from the left by v^H and using (3.7b),

$$\lambda \|D(X)v\|_2^2 = v^H D(X)^T D(Y) v = \bar{\lambda}^{-1} \|D(Y)v\|_2^2$$

which implies

$$|\lambda|^2 \|D(X)v\|_2^2 = \|D(Y)v\|_2^2. \tag{3.8}$$

Now, we decompose $D(Y)v$ orthogonally as

$$D(Y)v = cD(X)v + w, \tag{3.9}$$

with $v^H D(X)^T w = 0$. Substituting (3.9) into (3.7a) and multiplying both sides from the left by v^H yields

$$\lambda v^H D(X)^T D(X) v = c v^H D(X)^T D(X) v.$$

Since $v \neq 0$, and under Assumption 3.1.3, we deduce that $c = \lambda$. Substituting the value of c in (3.9), finding the 2-norm, and using the fact that $v^H D(X)^T w = 0$, one can write

$$\|D(Y)v\|_2^2 = |\lambda|^2 \|D(X)v\|_2^2 + \|w\|_2^2.$$

Comparing this with (3.8), one deduces that $w = 0$ and $D(Y)v = \lambda D(X)v$. The result follows by looking at this equality in a row-wise manner and noting that $f(\cdot) = D(\cdot)v$.

($\Rightarrow$): Based on Lemma 3.2.1(i), v must be an eigenvector of K_F with eigenvalue λ. Moreover, since $\lambda \neq 0$ one can write $f(x_i) = \lambda^{-1} f(y_i)$ for every $i \in \{1, \ldots, \sharp\text{rows}(X)\}$ and, consequently, using Lemma 3.2.1(i) once again, we have $K_B v = \lambda^{-1} v$, concluding the proof. $\square$

If the function f satisfies the conditions provided by Theorem 3.2.3, then $f(x^+) = \lambda f(x)$ for all $x \in \text{rows}(X)$. However, Theorem 3.2.3 does not guarantee that f is an eigenfunction, i.e., there is no guarantee that $f(x^+) = \lambda f(x)$ for all $x \in \mathcal{M}$. To circumvent this issue, we introduce an infinite sampling scheme and make an assumption about its density.

Assumption 3.2.4. ***(Almost sure dense sampling from a compact state space).*** *Assume the state space $\mathcal{M}$ is compact. Suppose we gather infinitely (countably) many data* *snapshots. For $N \in \mathbb{N}$, the first N data snapshots are represented by matrices $X_{1:N}$ and $Y_{1:N}$ such that $y_i = T(x_i)$ for all $i \in \{1, \dots, N\}$, where x_i and y_i are the ith rows of $X_{1:N}$ and $Y_{1:N}$, respectively (we refer to the columns of $X_{1:N}^T$ as the set S_N of initial conditions). Assume there exists a class-$\mathcal{K}$ function α and sequence $\{p_N\}_{N=1}^{\infty} \subset [0,1]$ such that, for every $N \in \mathbb{N}$,*

$$\forall m \in \mathcal{M}, \exists x \in S_N \text{ such that } \|m - x\|_2 \le \alpha\Big(\frac{1}{N}\Big)$$

holds with probability p_N, and $\lim_{N \to \infty} p_N = 1$. □

Assumption 3.2.4 is not restrictive as, in most practical cases, the state space is compact or the analysis is limited to a specific bounded region. Moreover, the data is usually available on a bounded region due the limited range of sensors. Regarding the sampling density, Assumption 3.2.4 holds for most standard random samplings.

Noting that our methods presented later require Assumption 3.1.3 to hold, we provide a definition for dictionary matrices acquired from infinite sampling.

Definition 3.2.5. ***(R-rich Sequence of Dictionary Snapshots).*** *Let $\{X_{1:N}\}_{N=1}^{\infty}$ and $\{Y_{1:N}\}_{N=1}^{\infty}$ be the sequence of data snapshot matrices acquired from system* (3.1). *Given the dictionary $D : \mathcal{M} \to \mathbb{R}^{1 \times N_d}$, we say the sequence of dictionary snapshot matrices is R-*rich *if $R = \min\{M \in \mathbb{N} \,|\, \mathrm{rank}(D(X_{1:M})) = \mathrm{rank}(D(Y_{1:M})) = N_d\}$ exists (R is called* richness constant*).* □

In Definition 3.2.5, if

$$\{M \in \mathbb{N} \,|\, \operatorname{rank}(D(X_{1:M})) = \operatorname{rank}(D(Y_{1:M})) = N_d\} \neq \emptyset$$

then based on the well-ordering principle, see e.g. [Fol99, Chapter 0], the minimum of the set exists and the sequence of the dictionary snapshot matrices is R-rich. Moreover, given an R-rich sequence of dictionary snapshots matrices $D(X_{1:N})$ and $D(Y_{1:N})$, Assumption 3.1.3 holds for every $N \geq R$.

We are now ready to identify the Koopman eigenfunctions in the span of the dictionary using forward-backward EDMD.

Theorem 3.2.6. ***(Identification of Koopman Eigenfunctions by Forward and Backward EDMD).*** *Given an infinite sampling, suppose that the sequence of dictionary snapshot matrices is R-rich. Let $K_{F,N} = \text{EDMD}(D, X_{1:N}, Y_{1:N})$, $K_{B,N} = \text{EDMD}(D, Y_{1:N}, X_{1:N})$. Given $v \in \mathbb{C}^{N_d} \setminus \{0\}$ and $\lambda \in \mathbb{C} \setminus \{0\}$, let $f(\cdot) = D(\cdot)v$. Then,*

(i) If f is an eigenfunction of the Koopman operator with eigenvalue λ, then $K_{F,N}v = \lambda v$ and $K_{B,N}v = \lambda^{-1}v$ for every $N \geq R$;

(ii) Conversely, and assuming the dictionary functions are continuous and Assumption 3.2.4 holds, if $K_{F,N}v = \lambda v$ and $K_{B,N}v = \lambda^{-1}v$ for every $N \geq R$, then f is an eigenfunction of the Koopman operator with probability 1.

Proof. (i) Since f is a Koopman eigenfunction, for every $i \in \mathbb{N}$ we have $f(y_i) = \mathcal{K}f(x_i) = \lambda f(x_i)$. Moreover, for every $N \geq R$, $D(X_{1:N})$ and $D(Y_{1:N})$ have full column rank. Therefore, the result follows from Theorem 3.2.3.

(ii) Based on Theorem 3.2.3, we deduce that, for every $N \geq R$

$$f(y_i) = \lambda f(x_i) v, \; \forall i \in \{1, \dots, N\}, \tag{3.10}$$

where x_i^T and y_i^T are the ith rows of $X_{1:N}$ and $Y_{1:N}$ respectively. Now, define $h(x) := \mathcal{K}f(x) - \lambda f(x) = f \circ T(x) - \lambda f(x)$. The function h is continuous since f is a linear combination of continuous functions and T is also continuous. By inspecting h on the data points and using (3.10) and the fact that $y_i = T(x_i)$, for all $i \in \{1, \dots, N\}$, one can show that $h(x_i) = f \circ T(x_i) - \lambda f(x_i) = f(y_i) - \lambda f(x_i) = 0$ for every $i \in \{1, \dots, N\}$. Moreover, note that based on Assumption 3.2.4, the set $S_\infty = \bigcup_{i=1}^{\infty} S_i$ is dense in $\mathcal{M}$ with probability 1 and $h(x) = 0$ for every $x \in S_\infty$. As a result, $h(x) = 0$ on $\mathcal{M}$ with probability 1. This implies that $f \circ T(x) = \lambda f(x)$ for every $x \in \mathcal{M}$ almost surely. Consequently, we have $\mathcal{K}f = \lambda f$ almost surely, and the result follows. $\square$

We note that the technique of considering the evolution forward and backward in time has also been used in the literature for other purposes, e.g., to alleviate the effect of measurement noise on the data when performing DMD [DHWR16,HRDC17]. To our knowledge, the use of this technique here for the identification of Koopman eigenfunctions and invariant subspaces is novel. Moreover, unlike [KM20, Algorithm 1], the methods proposed here do not require access to the system's multi-step trajectories. Theorems 3.2.3 and 3.2.6 provide conditions to identify Koopman eigenfunctions. The identified eigenfunctions then can span Koopman-invariant subspaces. However, one still needs to compare N_d potentially complex eigenvectors and their corresponding eigenvalues. This procedure can be impractical for large N_d. Moreover, since $\mathcal{M} \subseteq \mathbb{R}^n$, the eigenfunctions of the Koopman operator form complex-conjugate pairs. Such pairs can be fully characterized using their real and imaginary parts, which allows to use instead real-valued functions. This motivates the development of algorithms to directly identify Koopman-invariant subspaces.

3.3 Identifying Koopman-Invariant Subspaces via Symmetric Subspace Decomposition

Here we provide an algorithmic method to identify Koopman-invariant subspaces in the span of a predefined dictionary and later show how it can be used to find Koopman eigenfunctions. With the setup of Section 3.1, given the original dictionary $D : \mathcal{M} \to \mathbb{R}^{1\times N_d}$ comprised of N_d linearly independent functions, we aim to find a dictionary $\tilde{D} : \mathcal{M} \to \mathbb{R}^{1\times \tilde{N}_d}$ with $\tilde{N}_d$ linearly independent functions such that the elements of $\tilde{D}$ span the maximal Koopman-invariant subspace in $\operatorname{span}(D)$. Since the invariance of $\operatorname{span}(\tilde{D})$ reflects in data, given Assumption 3.1.3 we have

$$\mathcal{R}(\tilde{D}(Y)) = \mathcal{R}(\tilde{D}(X)). \tag{3.11}$$

Moreover, since the elements of $\tilde{D}$ are in the span of D, there exists a full column rank matrix C such that $\tilde{D}(\cdot) = D(\cdot)C$, for all $x \in \mathcal{M}$. Thus from (3.11),

$$\mathcal{R}(D(Y)C) = \mathcal{R}(D(X)C). \tag{3.12}$$

Hence, we can reformulate the problem as a purely linear-algebraic problem consisting of finding the full column rank matrix C with maximum number of columns such that (3.12) holds. To solve this problem, we propose the Symmetric Subspace Decomposition (SSD) method. The SSD algorithm relies on the fact, from (3.11), that

$$\mathcal{R}(\tilde{D}(Y)) = \mathcal{R}(\tilde{D}(X)) \subseteq \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y)).$$

This fact can alternatively be expressed using the null space of the concatenation $[D(X), D(Y)]$. SSD uses the null space to prune the dictionary and remove functions that do not evolve linearly in time according to the available data to identify a potentially smaller dictionary. At each iteration, SSD repeats the aforementioned procedure of (i) concatenation of current dictionary matrices, (ii) null space identification, and (iii) dictionary reduction, until the desired dictionary is identified. Algorithm 1 presents the pseudocode[1].

3.3.1 Convergence Analysis of the SSD Algorithm

Here, we characterize the convergence properties of the SSD algorithm. The next result characterizes the dimension, maximality, and symmetry of the subspace defined by its output.

Theorem 3.3.1. ***(Properties of SSD Output).*** *Suppose Assumption 3.1.3 holds. For matrices $D(X), D(Y)$, let $C_{\text{SSD}} = \text{SSD}\big(D(X), D(Y)\big)$. The SSD algorithm has the following properties:*

[1]The function null($[A_i, B_i]$) returns a basis for the null space of $[A_i, B_i]$, and Z_i^A and Z_i^B in Step 4 have the same size.

Algorithm 1 Symmetric Subspace Decomposition

```
    Inputs: $D(X), D(Y) \in \mathbb{R}^{N \times N_d}$    Output: $C_{\text{SSD}}$
    Procedure: $C_{\text{SSD}} \leftarrow \text{SSD}\big(D(X), D(Y)\big)$
 1: Initialization
 2: $i \leftarrow 1$, $A_1 \leftarrow D(X)$, $B_1 \leftarrow D(Y)$, $C_{\text{SSD}} \leftarrow I_{N_d}$
 3: while 1 do
 4:     $\begin{bmatrix} Z_i^A \\ Z_i^B \end{bmatrix} \leftarrow \text{null}([A_i, B_i])$        ▷ Basis for the null space
 5:     if $\text{null}([A_i, B_i]) = \emptyset$ then
 6:         return 0                                          ▷ The basis does not exist
 7:         break
 8:     end if
 9:     if $\sharp\text{rows}(Z_i^A) \leq \sharp\text{cols}(Z_i^A)$ then
10:         return $C_{\text{SSD}}$                            ▷ The procedure is complete
11:         break
12:     end if
13:     $C_{\text{SSD}} \leftarrow C_{\text{SSD}} Z_i^A$         ▷ Reducing the subspace
14:     $A_{i+1} \leftarrow A_i Z_i^A$, $B_{i+1} \leftarrow B_i Z_i^A$, $i \leftarrow i + 1$
15: end while
```

(i) it stops after at most N_d iterations;

(ii) the matrix C_{SSD} is either 0 or has full column rank, and satisfies $\mathcal{R}(D(X)C_{\text{SSD}}) = \mathcal{R}(D(Y)C_{\text{SSD}})$;

(iii) the subspace $\mathcal{R}(D(X)C_{\text{SSD}})$ is maximal, in the sense that, for any matrix E with $\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E)$, we have $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C_{\text{SSD}})$ and $\mathcal{R}(E) \subseteq \mathcal{R}(C_{\text{SSD}})$;

(iv) $\mathcal{R}\big(\text{SSD}\big(D(X), D(Y)\big)\big) = \mathcal{R}\big(\text{SSD}\big(D(Y), D(X)\big)\big)$.

Proof. (i) First, we use (strong) induction to prove that at each iteration Z_i^A, Z_i^B are matrices with full column rank *upon existence*. By Assumption 3.1.3, A_1 and B_1 have full column rank. Now, by using Lemma 3.7.1 one can derive that Z_1^A and Z_1^B have full column rank. Now, suppose that the matrices $Z_1^A, \ldots, Z_k^A$ and $Z_1^B, \ldots, Z_k^B$ have full column rank. Using

Assumption 3.1.3 one can deduce that $A_{k+1} = A_1 Z_1^A \dots Z_k^A$, $B_{k+1} = B_1 Z_1^A \dots Z_k^A$ have full column rank since they are product of matrices with full column rank. Using Lemma 3.7.1, one can conclude that Z_{k+1}^A and Z_{k+1}^B have full column rank.

Consequently, we have $\sharp\text{rows}(Z_i^A) \geq \sharp\text{cols}(Z_i^A)$. Hence, Step 9 of the SSD algorithm implies that the algorithm can only move to the next iteration if $\sharp\text{rows}(Z_i^A) > \sharp\text{cols}(Z_i^A)$, which means the number of columns in A_{i+1} and B_{i+1} decreases with respect to A_i and B_i. Hence, the algorithm terminates after at most N_d iterations since A_1 and B_1 have N_d columns.

(ii) The $C_{\text{SSD}} = 0$ case is trivial. Suppose that the algorithm stops after k iterations with nonzero C_{SSD}. This means that Z_k^A and Z_k^B are square full rank matrices. Also, by definition we have $A_k Z_k^A = -B_k Z_k^B$ which means that $A_k = -B_k Z_k^B (Z_k^A)^{-1}$. Noting that $Z_k^B (Z_k^A)^{-1}$ is a full rank square matrix, one can derive $\mathcal{R}(A_k) = \mathcal{R}(B_k)$. A closer look at the definitions shows that $A_k = D(X) C_{\text{SSD}}$ and $B_k = D(Y) C_{\text{SSD}}$. Hence, $\mathcal{R}(D(X) C_{\text{SSD}}) = \mathcal{R}(D(Y) C_{\text{SSD}})$. Moreover, $C_{\text{SSD}} = Z_1^A \cdots Z_{k-1}^A$ and considering the fact that $Z_1^A, \dots, Z_{k-1}^A$ have full column rank, one can deduce that C_{SSD} has full column rank.

(iii) Suppose that the matrix E satisfies $\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E)$. First, we use induction to prove that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_i) \cap \mathcal{R}(B_i)$ for each iteration i that the algorithm goes through. Let $i = 1$, then $A_1 = D(X)$ and $B_1 = D(Y)$. Consequently, $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_1)$ and $\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E) \subseteq \mathcal{R}(B_1)$ based on the definition of E. Hence, $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_1) \cap \mathcal{R}(B_1)$. Now, suppose

$$\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_i) \cap \mathcal{R}(B_i). \tag{3.13}$$

Using Lemma 3.7.1, one can derive $\mathcal{R}(A_i Z_i^A) = \mathcal{R}(A_i) \cap \mathcal{R}(B_i)$. Combining this with (3.13), we get

$$\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_i Z_i^A) = \mathcal{R}\big(D(X) Z_1^A \cdots Z_i^A\big). \tag{3.14}$$

Using (3.14) with Lemma 3.7.2 one can derive $\mathcal{R}(E) \subseteq \mathcal{R}(Z_1^A \cdots Z_i^A)$. Using Lemma 3.7.2 once again, we get

$$\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E) \subseteq \mathcal{R}\big(D(Y) Z_1^A \cdots Z_i^A\big). \tag{3.15}$$

Definition of A_{i+1}, B_{i+1} along with (3.14) and (3.15) lead to conclusion that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_{i+1}) \cap \mathcal{R}(B_{i+1})$ and the induction is complete.

Now, suppose that the algorithm terminates at iteration k. In the case that $C_{\text{SSD}} = 0$, we have $\mathcal{R}(A_k) \cap \mathcal{R}(B_k) = \{0\}$, which means that $E = 0$ and $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C_{\text{SSD}})$. In the case that $C_{\text{SSD}} \neq 0$, using the fact that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_k) \cap \mathcal{R}(B_k)$, $C_{\text{SSD}} = Z_1^A \cdots Z_{k-1}^A$, and $\mathcal{R}(D(X)C_{\text{SSD}}) = \mathcal{R}(D(Y)C_{\text{SSD}})$, one can deduce that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C_{\text{SSD}})$. Moreover, using Assumption 3.1.3 and Lemma 3.7.2 one can write $\mathcal{R}(E) \subseteq \mathcal{R}(C_{\text{SSD}})$.

(iv) For convenience, let $E_{\text{SSD}} = \text{SSD}\big(D(Y), D(X)\big)$. Based on the definition of C_{SSD}

and E_{SSD}, one can write

$$\mathcal{R}(D(X)C_{\text{SSD}}) = \mathcal{R}(D(Y)C_{\text{SSD}})$$
$$\mathcal{R}(D(X)E_{\text{SSD}}) = \mathcal{R}(D(Y)E_{\text{SSD}})$$

These equations in conjunction with the maximality of $\mathcal{R}(C_{\text{SSD}})$ from part (iii) imply $\mathcal{R}(E_{\text{SSD}}) \subseteq \mathcal{R}(C_{\text{SSD}})$. Using a similar argument, invoking the maximality of $\mathcal{R}(E_{\text{SSD}})$, we have $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(E_{\text{SSD}})$, concluding the proof. □

Remark 3.3.2. ***(Time and Space Complexity of the SSD Algorithm).*** Given N data snapshots and a dictionary with N_d elements, where usually $N \gg N_d$, and assuming that operations on scalar elements require time and memory of order $O(1)$, the most time and memory consuming operation in the SSD algorithm is Step 4. This step can be done by truncated Singular Value Decomposition (SVD) and finding the perpendicular space to the span of the right singular vectors, with time complexity $O(NN_d^2)$ and memory complexity $O(NN_d)$, see e.g., [LWC19]. Since, based on Theorem 3.3.1(i), the SSD algorithm terminates in at most N_d iterations, the total time complexity is $O(NN_d^3)$. However, since at each iteration we can reuse the memory for Step 4, the space complexity of SSD is $O(NN_d)$. □

Note that SSD removes the functions that do not evolve linearly in time according to the available data snapshots. Therefore, as we gather more data, the identified subspace either remains the same or gets smaller, as stated next.

Lemma 3.3.3. ***(Monotonicity of SSD Output with Respect to Data Addition).***

Let $D(X), D(Y)$ and $D(\hat{X}), D(\hat{Y})$ be two pairs of data snapshots such that

$$\text{rows}\big([D(X), D(Y)]\big) \subseteq \text{rows}\big([D(\hat{X}), D(\hat{Y})]\big), \tag{3.16}$$

and for which Assumption 3.1.3 holds. Then

$$\mathcal{R}\big(\text{SSD}(D(\hat{X}), D(\hat{Y}))\big) \subseteq \mathcal{R}\big(\text{SSD}(D(X), D(Y))\big).$$

Proof. We use the shorthand notation $\hat{C} = \text{SSD}\big(D(\hat{X}), D(\hat{Y})\big)$ and $C = \text{SSD}\big(D(X), D(Y)\big)$. From (3.16), we deduce that there exists a matrix E with $\text{rows}(E) \subseteq \text{rows}(I_{\sharp\text{rows}(\hat{X})})$ such that

$$ED(\hat{X}) = D(X),\ ED(\hat{Y}) = D(Y). \tag{3.17}$$

Moreover, based on the definition of $\hat{C}$ and Theorem 3.3.1(ii), we have $\mathcal{R}(D(\hat{X})\hat{C}) = \mathcal{R}(D(\hat{Y})\hat{C})$. Hence, there exists a full rank square matrix $\hat{K}$ such that

$$D(\hat{Y})\hat{C} = D(\hat{X})\hat{C}\hat{K}.$$

Multiplying both sides from the left by E and using (3.17) gives $D(Y)\hat{C} = D(X)\hat{C}\hat{K}$. Consequently, we have $\mathcal{R}(D(Y)\hat{C}) = \mathcal{R}(D(X)\hat{C})$. Now, the maximality of C (Theorem 3.3.1(iii)) implies $\mathcal{R}(\hat{C}) \subseteq \mathcal{R}(C)$. □

Remark 3.3.4. ***(Implementing SSD on Finite-Precision Machines).*** Since SSD is iterative, its implementation using finite precision leads to small errors that can affect the rank and null space of $[A_i, B_i]$ in Step 4. To circumvent this issue, one can approximate $[A_i, B_i]$ at each iteration by a close (in the Frobenius norm) low-rank matrix. Let $\sigma_1 \geq \ldots \geq \sigma_{l_i}$ be the singular values of $[A_i, B_i] \in \mathbb{R}^{N \times l_i}$. Given a design parameter $\epsilon > 0$, let k_i be the minimum integer such that

$$\sum_{j=k_i}^{l_i} \sigma_j^2 \leq \epsilon \big(\sum_{j=1}^{l_i} \sigma_j^2 \big). \tag{3.18}$$

One can then construct the matrix $[\hat{A}_i, \hat{B}_i]$ by setting $\sigma_{k_i} = \cdots = \sigma_{l_i} = 0$ in the singular value decomposition of $[A_i, B_i]$. The resulting matrix has lower rank and

$$\|[A_i, B_i] - [\hat{A}_i, \hat{B}_i]\|_F^2 \leq \epsilon \|[A_i, B_i]\|_F^2. \tag{3.19}$$

Hence, ϵ tunes the accuracy of the approximation. It is important to note that similar error bounds can be found for other unitarily invariant norms, see e.g. [Mir60]. $\square$

3.3.2 Identification of Linear Evolutions and Koopman Eigenfunctions with the SSD Algorithm

Here, we study the properties of the output of the SSD algorithm in what concerns the identification of the maximal Koopman-invariant subspace and the Koopman eigenfunctions.

If $C_{\mathrm{SSD}} \neq 0$, we define the invariant dictionary as

$$D_{\mathrm{SSD}}(\cdot) := D(\cdot)C_{\mathrm{SSD}}. \tag{3.20}$$

To find the action of the Koopman operator on the subspace spanned by D_{SSD}, we apply EDMD on $\tilde{D}(X)$ and $\tilde{D}(Y)$ to find

$$\begin{aligned} K_{\mathrm{SSD}} &= \mathrm{EDMD}(D_{\mathrm{SSD}}, X, Y) = D_{\mathrm{SSD}}(X)^{\dagger} D_{\mathrm{SSD}}(Y) \\ &= \big(D(X)C_{\mathrm{SSD}}\big)^{\dagger}\big(D(Y)C_{\mathrm{SSD}}\big). \end{aligned} \tag{3.21}$$

Based on Theorem 3.3.1(ii), we have

$$\mathcal{R}(D_{\mathrm{SSD}}(X)) = \mathcal{R}(D_{\mathrm{SSD}}(Y)). \tag{3.22}$$

Moreover, $D_{\mathrm{SSD}}(X)$ and $D_{\mathrm{SSD}}(Y)$ have full column rank as a result of Assumption 3.1.3 and Theorem 3.3.1(ii). Consequently, K_{SSD} is a (unique) nonsingular matrix satisfying

$$D_{\mathrm{SSD}}(Y) = D_{\mathrm{SSD}}(X)K_{\mathrm{SSD}}. \tag{3.23}$$

Interestingly, equation (3.23) implies that the residual error of EDMD, $\|\tilde{D}(Y) - \tilde{D}(X)K_{\mathrm{SSD}}\|_F$, is equal to zero. Based on (3.22), one can find K_{SSD} more efficiently and only based on partial data instead of calculating the pseudo-inverse of $D(X)C_{\mathrm{SSD}}$. Formally, consider full column rank data matrices $\hat{X}, \hat{Y}$ such that

$$\operatorname{rows}\big([\hat{X}, \hat{Y}]\big) \subseteq \operatorname{rows}\big([X, Y]\big).$$

Then, $K_{\mathrm{SSD}} = \mathrm{EDMD}(D_{\mathrm{SSD}}, \hat{X}, \hat{Y})$. Next, we show that the eigenvectors of K_{SSD} fully characterize the functions that evolve linearly in time according to the available data.

Theorem 3.3.5. ***(Identification of Linear Evolutions using the SSD Algorithm).*** *Suppose that Assumption 3.1.3 holds. Let* $C_{\mathrm{SSD}} = \mathrm{SSD}\big(D(X), D(Y)\big) \neq 0$, $K_{\mathrm{SSD}} = \big(D(X)C_{\mathrm{SSD}}\big)^{\dagger}\big(D(Y)C_{\mathrm{SSD}}\big)$, *and* $f \in \operatorname{span}(D)$ *denoted as* $f(\cdot) = D(\cdot)v$ *with* $v \in \mathbb{C}^{N_d} \setminus \{0\}$. *Then* $\mathcal{K}f(x_i) = f(y_i) = \lambda f(x_i)$ *for some* $\lambda \in \mathbb{C} \setminus \{0\}$ *and for all* $i \in \{1, \ldots, \sharp\mathrm{rows}(X)\}$ *if and only if* $v = C_{\mathrm{SSD}}w$ *with* $K_{\mathrm{SSD}}w = \lambda w$.

Proof. ($\Leftarrow$): Based on definition of K_{SSD}, Assumption 3.1.3, and considering the fact that C_{SSD} has full column rank (Theorem 3.3.1(ii)), one can use (3.20)-(3.23) and the fact that $K_{\mathrm{SSD}}w = \lambda w$ to write $D(Y)C_{\mathrm{SSD}}w = \lambda D(X)C_{\mathrm{SSD}}w$. Consequently, using $v = C_{\mathrm{SSD}}w$ we have

$$D(Y)v = \lambda D(X)v.$$

By inspecting the equation above in a row-wise manner, one can deduce that $f(y_i) = \lambda f(x_i)$ for some $\lambda \in \mathbb{C} \setminus \{0\}$ and for all $i \in \{1, \ldots, \sharp\mathrm{rows}(X)\}$, as claimed.

($\Rightarrow$): Based on the hypotheses, we have

$$D(Y)v = \lambda D(X)v. \tag{3.24}$$

Consider first the case when $v \in \mathbb{R}^{N_d}$. Then using (3.24), we deduce $\mathcal{R}(D(X)v) = \mathcal{R}(D(Y)v)$. The maximality of C_{SSD} (Theorem 3.3.1(iii)) implies that $\mathcal{R}(v) \subseteq \mathcal{R}(C_{\mathrm{SSD}})$ and consequently $v = C_{\mathrm{SSD}}w$ for some w. Replacing v by $C_{\mathrm{SSD}}w$ in (3.24) and using the definition of K_{SSD}, one deduces $K_{\mathrm{SSD}}w = \lambda w$.

Now, suppose that $v = v_R + jv_I$ with $v_I \neq 0$. Since $D(X)$ and $D(Y)$ are real matrices, one can use (3.24) and write $D(Y)\bar{v} = \bar{\lambda}D(X)\bar{v}$. This, together with (3.24), implies

$$D(Y)E = D(X)E\Lambda, \tag{3.25}$$

where $E = [v_R, v_I]$ and

$$\Lambda = \begin{bmatrix} \mathrm{Re}(\lambda) & \mathrm{Im}(\lambda) \\ -\mathrm{Im}(\lambda) & \mathrm{Re}(\lambda) \end{bmatrix}.$$

Since Λ is full rank, we have $\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E)$ and using Theorem 3.3.1(iii), one can conclude $\mathcal{R}(E) \subseteq \mathcal{R}(C_{\mathrm{SSD}})$. Consequently, there exists a real vector z such that $E = C_{\mathrm{SSD}}z$. By replacing this in (3.25) and multiplying both sides from the right by $r = [1, j]^T$ and defining $w = zr$, one can conclude that $v = Er = C_{\mathrm{SSD}}w$ and $D(Y)C_{\mathrm{SSD}}w = \lambda D(X)C_{\mathrm{SSD}}w$. This in conjunction with the definition of K_{SSD} implies that $K_{\mathrm{SSD}}w = \lambda w$, concluding the proof. $\square$

Using Theorem 3.3.5, one can identify all the linear evolutions in the span of the original dictionary, thereby establishing an equivalence with the forward-backward EDMD characterization of Section 3.2.

Corollary 3.3.6. ***(Equivalence of Forward-Backward EDMD and SSD in the Identification of Linear Evolutions).*** *Suppose that Assumption 3.1.3 holds. Let* $K_F = \text{EDMD}(D, X, Y)$*,* $K_B = \text{EDMD}(D, Y, X)$*,* $C_{\text{SSD}} = \text{SSD}\big(D(X), D(Y)\big) \neq 0$ *and* $K_{\text{SSD}} = \big(D(X)C_{\text{SSD}}\big)^{\dagger}\big(D(Y)C_{\text{SSD}}\big)$*. Then,* $K_F v = \lambda v$ *and* $K_B v = \lambda^{-1} v$ *for some* $v \in \mathbb{C}^{N_d} \setminus \{0\}$ *and* $\lambda \in \mathbb{C} \setminus \{0\}$ *if and only if there exists vector* w *such that* $v = C_{\text{SSD}} w$ *and* $K_{\text{SSD}} w = \lambda w$*.*

The proof of this result is a consequence of Theorems 3.2.3 and 3.3.5. Note that the linear evolutions identified by SSD might not be Koopman eigenfunctions, since we can only guarantee that they evolve linearly according to the available data snapshots, not starting everywhere in the state space $\mathcal{M}$. The following result uses the equivalence between SSD and the Forward-Backward EDMD method to provide a guarantee for the identification of Koopman eigenfunctions.

Theorem 3.3.7. ***(Identification of Koopman Eigenfunctions by the SSD Algorithm).*** *Given an infinite sampling, suppose that the sequence of dictionary snapshot matrices is R-rich. For* $N \geq R$*, let* $C_{\text{SSD},N} = \text{SSD}\big(D(X_{1:N}), D(Y_{1:N})\big) \neq 0$*, and* $K_{\text{SSD},N} = \big(D(X_{1:N})C_{\text{SSD},N}\big)^{\dagger}\big(D(Y_{1:N})C_{\text{SSD},N}\big)$*. Given* $v \in \mathbb{C}^{N_d} \setminus \{0\}$ *and* $\lambda \in \mathbb{C} \setminus \{0\}$*, let* $f(\cdot) = D(\cdot)v$*. Then,*

(i) If f *is an eigenfunction of the Koopman operator with eigenvalue* λ*, then for every* $N \geq R$*, there exists* w_N *such that* $v = C_{\text{SSD},N} w_N$ *and* $K_N^{\text{SSD}} w_N = \lambda w_N$*;*

(ii) Conversely, and assuming the dictionary functions are continuous and Assumption 3.2.4 holds, if $v \in \mathcal{R}(C_{\mathrm{SSD},N})$ *and there exists* w_N *such that* $v = C_{\mathrm{SSD},N} w_N$ *and* $K_{\mathrm{SSD},N} w_N = \lambda w_N$ *for every* $N \geq R$*, then* f *is an eigenfunction of the Koopman operator with probability 1.*

This result is a consequence of Theorem 3.2.6 and Corollary 3.3.6. Theorem 3.3.7 shows that the SSD algorithm finds all the eigenfunctions in the span of the original dictionary almost surely. The identified eigenfunctions span a Koopman-invariant subspace. This subspace however is not necessarily the maximal Koopman-invariant subspace in the span of the original dictionary. Next, we show that the SSD method actually identifies the maximal Koopman-invariant subspace in the span of the dictionary.

Theorem 3.3.8. ***(SSD Finds the Maximal Koopman-Invariant Subspace as*** $N \to \infty$***).*** *Given an infinite sampling and a dictionary composed of continuous functions, suppose that the sequence of dictionary snapshot matrices is R-rich and Assumption 3.2.4 holds. Let the columns of* $C_{\mathrm{SSD},\infty}$ *form a basis for* $\lim_{N\to\infty} \mathcal{R}(C_{\mathrm{SSD},N})$*, i.e.,*

$$\mathcal{R}(C_{\mathrm{SSD},\infty}) = \lim_{N\to\infty} \mathcal{R}(C_{\mathrm{SSD},N}) = \bigcap_{N=R}^{\infty} \mathcal{R}(C_{\mathrm{SSD},N}). \tag{3.26}$$

(note that the sequence $\{\mathcal{R}(C_{\mathrm{SSD},N})\}_{N=1}^{\infty}$ *is monotonic, and hence convergent). Then* $\operatorname{span}(D(\cdot)C_{\mathrm{SSD},\infty})$ *is the maximal Koopman-invariant subspace in the span of the dictionary* D *with probability 1.*

Proof. If $C_{\mathrm{SSD},\infty} = 0$, considering the fact that for all $N \geq R$, $\mathcal{R}(C_{\mathrm{SSD},N+1}) \subseteq \mathcal{R}(C_{\mathrm{SSD},N})$ (Lemma 3.3.3), one deduces that there exists $m \in \mathbb{N}$ such that for all $i \geq m$, $C_{\mathrm{SSD},i} = 0$.

Hence based on Theorem 3.3.1(iii), the maximal Koopman-invariant subspace acquired from the data is $\{0\}$. Noting that the subspace identified by SSD contains the maximal Koopman-invariant subspace, we deduce that the latter is the zero subspace, which is indeed spanned by $D(\cdot)C_{\mathrm{SSD},\infty}$.

Now, suppose that $C_{\mathrm{SSD},\infty} \neq 0$ and has full column rank. First, we show that

$$\mathcal{R}(D(X_{1:N})C_{\mathrm{SSD},\infty}) = \mathcal{R}(D(Y_{1:N})C_{\mathrm{SSD},\infty}), \quad \forall N \geq R. \tag{3.27}$$

Considering (3.26) and the fact that for all $N \geq R$, $\mathcal{R}(C_{\mathrm{SSD},N+1}) \subseteq \mathcal{R}(C_{\mathrm{SSD},N})$, we can write for all $N \geq R$

$$\mathcal{R}(C_{\mathrm{SSD},\infty}) = \bigcap_{i=N}^{\infty} \mathcal{R}(C_{\mathrm{SSD},i}).$$

Invoking Lemma 3.7.4, we have for all $N \geq R$,

$$\mathcal{R}(D(X_{1:N})C_{\mathrm{SSD},\infty}) = \bigcap_{i=N}^{\infty} \mathcal{R}(D(X_{1:N})C_{\mathrm{SSD},i}), \tag{3.28a}$$

$$\mathcal{R}(D(Y_{1:N})C_{\mathrm{SSD},\infty}) = \bigcap_{i=N}^{\infty} \mathcal{R}(D(Y_{1:N})C_{\mathrm{SSD},i}). \tag{3.28b}$$

Moreover, for all $i \geq N$ we have $\mathcal{R}(D(X_{1:i})C_{\mathrm{SSD},i}) = \mathcal{R}(D(Y_{1:i})C_{\mathrm{SSD},i})$ and hence by looking at this equality in a row-wise manner, one can write

$$\mathcal{R}(D(X_{1:N})C_{\mathrm{SSD},i}) = \mathcal{R}(D(Y_{1:N})C_{\mathrm{SSD},i}), \quad \forall i \geq N. \tag{3.29}$$

The combination of (3.28) and (3.29) yields (3.27). Based on the latter, the fact that $D(X_{1:N})$ and $D(Y_{1:N})$ have full column rank for every $N \geq R$ and the fact that $C_{\mathrm{SSD},\infty}$ has full column rank, there exists a *unique* nonsingular square matrix $K_{\mathrm{SSD},\infty} \in \mathbb{R}^{\sharp \mathrm{cols}(C_{\mathrm{SSD},\infty}) \times \sharp \mathrm{cols}(C_{\mathrm{SSD},\infty})}$ such that

$$D(X_{1:N}) C_{\mathrm{SSD},\infty} K_{\mathrm{SSD},\infty} = D(Y_{1:N}) C_{\mathrm{SSD},\infty}, \quad \forall N \geq R. \tag{3.30}$$

Note that $K_{\mathrm{SSD},\infty}$ does not depend on N. Next, we aim to prove that for every function $f \in \operatorname{span}(D(\cdot) C_{\mathrm{SSD},\infty})$, $\mathcal{K} f$ is also in $\operatorname{span}(D(\cdot) C_{\mathrm{SSD},\infty})$ almost surely. Let $v \in \mathbb{R}^{\sharp \mathrm{cols}(C_{\mathrm{SSD},\infty})}$ such that $f(\cdot) = D(\cdot) C_{\mathrm{SSD},\infty} v$ and define

$$g(\cdot) := D(\cdot) C_{\mathrm{SSD},\infty} K_{\mathrm{SSD},\infty} v. \tag{3.31}$$

We show that $g = f \circ T = \mathcal{K} f$ almost surely. Define the function $h := g - \mathcal{K} f = g - f \circ T$. Also, let $S_\infty = \bigcup_{N=R}^{\infty} S_N$ be the set of initial conditions. Based on (3.30), (3.31), and definition of h,

$$h(x) = 0, \quad \forall x \in S_\infty.$$

Moreover, h is continuous since D and T are continuous. This, together with the fact that S_∞ is dense in $\mathcal{M}$ almost surely (Assumption 3.2.4), we deduce $h \equiv 0$ on $\mathcal{M}$ almost surely. Therefore, $g = \mathcal{K} f = f \circ T$ with probability 1. Noting that $g(\cdot) \in \operatorname{span}(D(\cdot) C_{\mathrm{SSD},\infty})$, we have proven that $\operatorname{span}(D(\cdot) C_{\mathrm{SSD},\infty})$ is Koopman invariant almost surely.

Finally, we prove the maximality of $\mathrm{span}(D(\cdot)C_{\mathrm{SSD},\infty})$. Let $\mathcal{L}$ be a Koopman-invariant subspace in $\mathrm{span}(D)$. Then there exists a full column rank matrix E such that $\mathcal{L} = \mathrm{span}(D(\cdot)E)$. Moreover, since the invariance of $\mathcal{L}$ reflects in data, $\mathcal{R}(D(X_{1:N})E) = \mathcal{R}(D(Y_{1:N})E)$, for all $N \geq R$. As a result, based on Theorem 3.3.1(iii), we have $\mathcal{R}(E) \subseteq \mathcal{R}(C_{\mathrm{SSD},N})$, for all $N \geq R$, and hence $\mathcal{R}(E) \subseteq \mathcal{R}(C_{\mathrm{SSD},\infty})$. Therefore, by Lemma 3.7.2, we have $\mathcal{L} = \mathrm{span}(D(\cdot)E) \subseteq \mathrm{span}(D(\cdot)C_{\mathrm{SSD},\infty})$, which completes the proof. $\square$

Remark 3.3.9. ***(Generalized Koopman Eigenfunctions).*** One can also extend the above discussion for generalized Koopman eigenfunctions (see e.g. [BMM12, Remark 11]). Given a generalized eigenvector w of K_{SSD}, the corresponding generalized Koopman eigenfunction is $\phi(\cdot) = D(\cdot)C_{\mathrm{SSD}}w$. $\square$

3.4 Streaming Symmetric Subspace Decomposition

In this section, we consider the setup where data becomes available in a streaming fashion. A straightforward algorithmic solution for this scenario would be to re-run, at each timestep, the SSD algorithm with all the data available up to then. However, this approach does not take advantage of the answers computed in previous timesteps, and may become inefficient when the size of the data is large. Instead, here we pursue the design of an online algorithm, termed Streaming Symmetric Subspace Decomposition (SSSD), cf. Algorithm 2, that updates the identified subspaces using the previously computed ones. Note that the SSSD algorithm is not only useful for streaming data sets but also for the case of non-streaming large data sets for which the execution of SSD requires a significant amount of memory.

Algorithm 2 Streaming Symmetric Subspace Decomposition

```
 1: Initialization
 2: $D_S^X(1) \leftarrow \begin{bmatrix} D(X_{1:S}) \\ D(x_{S+1}) \end{bmatrix}$, $D_S^Y(1) \leftarrow \begin{bmatrix} D(Y_{1:S}) \\ D(y_{S+1}) \end{bmatrix}$
 3: $i \leftarrow 1$, $A_1 \leftarrow D_S^X(1)$, $B_1 \leftarrow D_S^Y(1)$, $C_0 \leftarrow I_{N_d}$
 4: while 1 do
 5:     if $C_{i-1} = 0$ then
 6:         $C_i \leftarrow 0$                                  ▷ The basis does not exist
 7:         return $C_i$
 8:         break
 9:     end if
10:     $F_i \leftarrow \text{SSD}(A_i, B_i)$                           ▷ Using Algorithm 1
11:     if $F_i = 0$ then
12:         $C_i \leftarrow 0$                                  ▷ The basis does not exist
13:         return $C_i$
14:         break
15:     end if
16:     if $\sharp\text{rows}(F_i) > \sharp\text{cols}(F_i)$ then
17:         $C_i \leftarrow \text{basis}(\mathcal{R}(C_{i-1}F_i))$               ▷ Subspace reduction
18:     else
19:         $C_i \leftarrow C_{i-1}$                                     ▷ No change
20:     end if
21:     return $C_i$
22:     $i \leftarrow i + 1$
        ▽ Replacing the last data snapshot with the new one
23:     $D_S^X(i) = \begin{bmatrix} D(X_{1:S}) \\ D(x_{S+i}) \end{bmatrix}$, $D_S^Y(i) = \begin{bmatrix} D(Y_{1:S}) \\ D(y_{S+i}) \end{bmatrix}$
        ▽ Calculating the reduced dictionary snapshots
24:     $A_i \leftarrow D_S^X(i)C_{i-1}$, $B_i \leftarrow D_S^Y(i)C_{i-1}$
25: end while
```

Given the *signature* snapshot matrices $X_{1:S}$ and $Y_{1:S}$, for some $S \in \mathbb{N}$, and a dictionary of functions D, the SSSD algorithm proceeds as follows: at each iteration, the algorithm receives a new pair of data snapshots, combines them with signature data matrices, and applies the latest available dictionary on them. Then, it uses SSD on those dictionary matrices and further prunes the dictionary. The basic idea of the SSSD algorithm stems from the monotonicity of SSD's output dictionary versus the data (cf. Lemma 3.3.3), i.e., by adding more data the dimension of the dictionary does not increase. Since the SSD algorithm relies on Assumption 3.1.3, we make the following assumption on the signature snapshots and the original dictionary.

Assumption 3.4.1. ***(Full Rank Signature Dictionary Matrices).*** *We assume that there exists $S \in \mathbb{N}$ such that the matrices $D(X_{1:S})$ and $D(Y_{1:S})$ have full column rank.* $\square$

For a finite number of data snapshots, Assumption 3.4.1 is equivalent to Assumption 3.1.3. For an infinite sampling, Assumption 3.4.1 holds for an R-rich sequence of snapshot matrices. The next result discusses the basic properties of the SSSD output at each iteration.

Proposition 3.4.2. ***(Properties of SSSD Output).*** *Suppose Assumption 3.4.1 holds. For $i \in \mathbb{N}$, let C_i denote the output of the SSSD algorithm at the ith iteration. Then, for all $i \in \mathbb{N}$,*

(i) C_i has full column rank or is equal to zero;

(ii) $\mathcal{R}(C_i) \subseteq \mathcal{R}(C_{i-1})$;

(iii) $\mathcal{R}(D_S^X(i)C_i) = \mathcal{R}(D_S^Y(i)C_i)$.

Proof. (i) We prove the claim by induction. $C_0 = I_{N_d}$ and has full column rank. Now, suppose that C_k has full column rank or is zero. We show the same fact for C_{k+1}. If $C_k = 0$, then SSSD executes Step 6 and we have $C_{k+1} = 0$. Now, suppose that C_k has full column rank. Considering the fact that $D_S^X(k+1)$ and $D_S^Y(k+1)$ have full column rank, one can deduce that A_{k+1} and B_{k+1} have full column rank. Consequently, based on Theorem 3.3.1(ii), F_{k+1} has full column rank or is equal to zero. In the former case, the algorithm executes Step 17 or Step 19, and based on definition of basis function and the fact that C_k has full column rank, one deduces that C_{k+1} has full column rank. In the latter case, the algorithm executes Step 12, and $C_{k+1} = 0$, as claimed.

Now we prove (ii). Note that at iteration i, C_i will be determined by either Step 6, 12, 19, or 17. The proof for the first three cases is trivial. We only need to prove the result for the case when the SSSD algorithm executes Step 17. Based on Theorem 3.3.1(ii), one can deduce that F_i has full column rank. Also, we have $\mathcal{R}(F_i) \subseteq \mathcal{R}(I_{\sharp\text{cols}(C_{i-1})})$. Hence using Step 17 and Lemma 3.7.2, one can write

$$\mathcal{R}(C_i) = \mathcal{R}(C_{i-1}F_i) \subseteq \mathcal{R}(C_{i-1}I_{\sharp\text{cols}(C_{i-1})}) = \mathcal{R}(C_{i-1}),$$

as claimed.

Next, we prove part (iii). If the SSSD algorithm executes Step 6 or Step 12, then the result follows directly. Now, suppose that the algorithm executes Step 17 or Step 19. Note that if the algorithm executes one of these two steps, then $F_i \neq 0$, $C_{i-1} \neq 0$ and they have full column rank (Theorem 3.3.1(ii)). Hence, $\sharp\text{rows}(F_i) \geq \sharp\text{cols}(F_i)$. As a result, if the algorithm executes Step 19, we have $\sharp\text{rows}(F_i) = \sharp\text{cols}(F_i)$ and consequently $\mathcal{R}(F_i) = \mathcal{R}(I_{\sharp\text{cols}(C_{i-1})})$. Therefore,

$$\mathcal{R}(C_i) = \mathcal{R}(C_{i-1}) = \mathcal{R}(C_{i-1}I_{\sharp\text{cols}(C_{i-1})}) = \mathcal{R}(C_{i-1}F_i). \tag{3.32}$$

Moreover, if the SSSD algorithm executes Step 17, then using the definition of basis function, we have

$$\mathcal{R}(C_i) = \mathcal{R}(C_{i-1}F_i). \tag{3.33}$$

Also, based on definition of F_i at Step 10, Theorem 3.3.1(ii), and the fact that $A_i = D_S^X(i)C_{i-1}$ and $B_i = D_S^Y(i)C_{i-1}$,

$$\mathcal{R}(D_S^X(i)C_{i-1}F_i) = \mathcal{R}(D_S^Y(i)C_{i-1}F_i).$$

Using this together with (3.32) upon execution of Step 19 and (3.33) upon execution of Step 17, one deduces $\mathcal{R}(D_S^X(i)C_i) = \mathcal{R}(D_S^Y(i)C_i)$, concluding the proof. □

Next, we show that the SSSD algorithm at each iteration identifies exactly the same subspace as the SSD algorithm given all the data up to that iteration.

Theorem 3.4.3. ***(Equivalence of SSD and SSSD).*** *Suppose Assumption 3.4.1 holds. For $i \in \mathbb{N}$, let C_i denote the output of the SSSD algorithm at the ith iteration and let $C_{\mathrm{SSD},i} = \mathrm{SSD}\big(D(X_{1:S+i}), D(Y_{1:S+i})\big)$. Then,*

$$\mathcal{R}(C_i) = \mathcal{R}(C_{\mathrm{SSD},i}), \quad \forall i \in \mathbb{N}.$$

Proof. Inclusion $\mathcal{R}(C_{\mathrm{SSD},i}) \subseteq \mathcal{R}(C_i)$ for all $i \in \mathbb{N}$: We reason by induction. Note that in the SSSD algorithm, for $i = 1$ we have $F_1 = C_{\mathrm{SSD},1}$. As a result, if $F_1 = 0$ then based on Step 12, $C_1 = C_{\mathrm{SSD},1} = 0$. If the SSSD algorithm executes Step 17, then using the fact that

$C_0 = I_{N_d}$, one can write $\mathcal{R}(C_1) = \mathcal{R}(C_{\text{SSD},1})$. Moreover, if the SSSD algorithm executes Step 19, based on Step 16 and Theorem 3.3.1(ii), one can deduce that $\mathcal{R}(C_{\text{SSD},1}) = \mathcal{R}(C_1) = \mathcal{R}(F_1) = \mathcal{R}(I_{N_d})$. Consequently, in all cases

$$\mathcal{R}(C_{\text{SSD},1}) = \mathcal{R}(C_1). \tag{3.34}$$

Hence, $\mathcal{R}(C_{\text{SSD},1}) \subseteq \mathcal{R}(C_1)$. Now, suppose that

$$\mathcal{R}(C_{\text{SSD},k}) \subseteq \mathcal{R}(C_k). \tag{3.35}$$

We need to show that $\mathcal{R}(C_{\text{SSD},k+1}) \subseteq \mathcal{R}(C_{k+1})$. If $C_{\text{SSD},k+1} = 0$ then the proof follows. Now assume that $C_{\text{SSD},k+1} \neq 0$ and has full column rank based on Theorem 3.3.1(ii). By Lemma 3.3.3, we have

$$\mathcal{R}(C_{\text{SSD},k+1}) \subseteq \mathcal{R}(C_{\text{SSD},k}). \tag{3.36}$$

Using (3.35) and (3.36), one can deduce $\mathcal{R}(C_{\text{SSD},k+1}) \subseteq \mathcal{R}(C_k)$. Consequently, based on the fact that $C_{\text{SSD},k+1} \neq 0$, we have $C_k \neq 0$ and hence has full column rank based on Proposition 3.4.2(i). Moreover, there exists a full column-rank matrix E_k such that

$$C_{\text{SSD},k+1} = C_k E_k. \tag{3.37}$$

Two cases are possible. In case 1, the SSSD algorithm executes Step 19. In case 2, the algorithm executes Step 12 or Step 17. For case 1, we have $C_{k+1} = C_k$. Consequently, using (3.37) and considering the fact that $\mathcal{R}(E_k) \subseteq \mathcal{R}(I_{\sharp\mathrm{cols}(C_k)})$ and the fact that C_k has full column rank, one can use Lemma 3.7.2 and conclude

$$\mathcal{R}(C_{\mathrm{SSD},k+1}) = \mathcal{R}(C_k E_k) \subseteq \mathcal{R}(C_k) = \mathcal{R}(C_{k+1}). \tag{3.38}$$

Now, consider case 2. In this case, we have

$$\mathcal{R}(C_{k+1}) = \mathcal{R}(C_k F_{k+1}). \tag{3.39}$$

Also, based on definition of $C_{\mathrm{SSD},k+1}$ and Theorem 3.3.1(ii), one can write

$$\mathcal{R}(D(X_{1:k+1} C_{\mathrm{SSD},k+1})) = \mathcal{R}(D(Y_{1:k+1} C_{\mathrm{SSD},k+1})).$$

Looking at this equation in a row-wise manner and considering the fact that $\mathrm{rows}\big([D_S^X(k+1), D_S^Y(k+1)]\big) \subseteq \mathrm{rows}\big([D(X_{1:k+1}), D(Y_{1:k+1})]\big)$, one can write

$$\mathcal{R}(D_S^X(k+1) C_{\mathrm{SSD},k+1}) = \mathcal{R}(D_S^Y(k+1) C_{\mathrm{SSD},k+1}).$$

Now, using (3.37) we have $\mathcal{R}(D_S^X(k+1) C_k E_k) = \mathcal{R}(D_S^Y(k+1) C_k E_k)$. Also, noting the definition of F_{k+1} and the fact that $A_k = D_S^X(k+1) C_k$, $B_k = D_S^Y(k+1) C_k$, one can use Theorem 3.3.1(iii) to write $\mathcal{R}(E_k) \subseteq \mathcal{R}(F_{k+1})$. Since C_k has full column rank, we use (3.37), (3.39), and Lemma 3.7.2 to write

$$\mathcal{R}(C_{\text{SSD},k+1}) = \mathcal{R}(C_k E_k) \subseteq \mathcal{R}(C_k F_{k+1}) = \mathcal{R}(C_{k+1}). \tag{3.40}$$

In both cases, equations (3.38) and (3.40) conclude the induction.

Inclusion $\mathcal{R}(C_i) \subseteq \mathcal{R}(C_{\text{SSD},i})$ for all $i \in \mathbb{N}$: We reason by induction too. Using (3.34), we have $\mathcal{R}(C_1) \subseteq \mathcal{R}(C_{\text{SSD},1})$. Now, suppose that

$$\mathcal{R}(C_k) \subseteq \mathcal{R}(C_{\text{SSD},k}). \tag{3.41}$$

We prove the same result for $k+1$. If $C_{k+1} = 0$ then the result directly follows. Now, assume that $C_{k+1} \neq 0$. Consequently, based on (3.41), Proposition 3.4.2(i), and Theorem 3.3.1(ii), we deduce that C_{k+1} and $C_{\text{SSD},k+1}$ have full column rank.

The first part of the proof and (3.41) imply that $\mathcal{R}(C_k) = \mathcal{R}(C_{\text{SSD},k})$. Consequently, noting the fact that $C_{\text{SSD},k}$ is the output of the SSD algorithm with $D(X_{1:S+k})$ and $D(Y_{1:S+k})$, one can use Theorem 3.3.1(ii) to write

$$\mathcal{R}\big(D(X_{1:S+k})C_k\big) = \mathcal{R}\big(D(Y_{1:S+k})C_k\big). \tag{3.42}$$

Moreover, based on Proposition 3.4.2(ii), we have $\mathcal{R}(C_{k+1}) \subseteq \mathcal{R}(C_k)$. Hence, since C_k and C_{k+1} have full column rank, there exists a matrix G_k with full column rank such that

$$C_{k+1} = C_k G_k. \tag{3.43}$$

Also, based on Proposition 3.4.2(iii) at iteration $k+1$ of the SSSD algorithm

$$\mathcal{R}(D_S^X(k+1)C_{k+1}) = \mathcal{R}(D_S^Y(k+1)C_{k+1}). \tag{3.44}$$

Consequently, based on (3.43) and (3.44), we have

$$\mathcal{R}(D(X_{1:S})C_kG_k) = \mathcal{R}(D(Y_{1:S})C_kG_k).$$

Using this equation together with (3.42) and Lemma 3.7.3,

$$\mathcal{R}\big(D(X_{1:S+k})C_kG_k\big) = \mathcal{R}\big(D(Y_{1:S+k})C_kG_k\big).$$

Moreover, using (3.43) one can write

$$\mathcal{R}\big(D(X_{1:S+k})C_{k+1}\big) = \mathcal{R}\big(D(Y_{1:S+k})C_{k+1}\big).$$

Hence, there exists a nonsingular square matrix K^* such that

$$D(X_{1:S+k})C_{k+1}K^* = D(Y_{1:S+k})C_{k+1}. \tag{3.45}$$

Also, based on (3.44) and noting that $D_S^X(k+1)$, $D_S^X(k+1)$, and C_{k+1} have full column rank, there exists a nonsingular square matrix K such that

$$D_S^X(k+1)C_{k+1}K = D_S^Y(k+1)C_{k+1}. \tag{3.46}$$

Using the first S rows of (3.45) and (3.46), one can write

$$D(X_{1:S})C_{k+1}K^* = D(Y_{1:S})C_{k+1},$$
$$D(X_{1:S})C_{k+1}K = D(Y_{1:S})C_{k+1}.$$

By subtracting the second equation from the first one, we get

$$D(X_{1:S})C_{k+1}(K^* - K) = 0.$$

Moreover, since $D(X_{1:S})C_{k+1}$ has full column rank, we deduce $K^* = K$. Using this together with (3.45) and (3.46) yields

$$\mathcal{R}\big(D(X_{1:S+k+1})C_{k+1}\big) = \mathcal{R}\big(D(Y_{1:S+k+1})C_{k+1}\big). \tag{3.47}$$

From (3.47), the definition of $C_{\text{SSD},k+1}$ and Theorem 3.3.1(iii), we deduce $\mathcal{R}(C_{k+1}) \subseteq \mathcal{R}(C_{\text{SSD},k+1})$, concluding the proof. $\square$

Theorem 3.4.3 establishes the equivalence between the SSSD and SSD algorithms. As a consequence, all results regarding the identification of Koopman-invariant subspaces and eigenfunctions presented in Section 3.3 are also valid for the output of the SSSD algorithm.

Remark 3.4.4. ***(Time and Space Complexity of the SSSD Algorithm).*** Given the first N data snapshots and a dictionary with N_d elements, with $N > S \geq N_d$, and assuming that operations on scalar elements require time and space of order $O(1)$, the most time and memory consuming operation in the SSSD algorithm is Step 10 invoking SSD. In this step, the most time consuming operation is performing SVD, with time complexity $O(SN_d^2)$ and space complexity of $O(SN_d)$, see e.g., [LWC19]. After having performed the first SVD, the ensuing ones result in a reduction of the dimension of the subspace. Therefore, the SSSD algorithm performs at most $N - S$ SVDs with no subspace reduction with overall time complexity $O(NSN_d^2)$ and at most N_d SVD operations with subspace reductions with overall time complexity $O(SN_d^3)$. Considering the fact that $N \geq N_d$, the complexity of the SSSD algorithm is $O(NSN_d^2)$. Moreover, in many real world applications $S = O(N_d)$ (in fact usually $S = N_d$), which reduces the time complexity of SSSD to $O(NN_d^3)$, which is the same complexity as SSD. Moreover, since we can reuse the space used in Step 10 at each iteration, and considering the fact that the space complexity of this step is $O(SN_d)$, we deduce that the space complexity of SSSD is $O(SN_d)$. This usually reduces to $O(N_d^2)$ since $S = O(N_d)$ in many real-world applications. □

Remark 3.4.5. ***(SSSD is More Stable and Runs Faster than SSD).*** The SSSD algorithm is more computationally stable than SSD, since it always works with matrices of size at most $(S+1) \times N_d$ while SSD works with matrices of size $N \times N_d$. Moreover, even though SSD and SSSD have the same time complexity, the SSSD algorithm run faster for two reasons. First, at each iteration of the SSSD algorithm, the dictionary gets smaller, which reduces the cost of computation for the remaining data snapshots. Second, the characterizations in Remarks 3.3.2 and 3.4.4 only consider the number of floating point operations for the time complexity and ignore the amount of time used for loading the data. SSSD needs to load significantly smaller data matrices, which leads to a considerable reduction in run time compared to SSD. □

3.5 Approximating Koopman-Invariant Subspaces

We note that, if the span of the original dictionary D does not contain any Koopman-invariant subspace, then the SSD algorithm returns the trivial solution, which does not result in any information about the behavior of the dynamical system. To circumvent this issue, here we propose a method to *approximate* Koopman-invariant subspaces. Noting the fact that the existence of a Koopman-invariant subspace translates into the rank deficiency of the concatenated matrix $[A_i, B_i]$ in Step 4 of the SSD algorithm (cf. 1), we propose to replace the null function in SSD with the approx-null routine presented in Algorithm 3 below. This routine constructs an approximated null space by selecting a set of small singular values. The parameter $\epsilon > 0$ in Algorithm 3 is a design choice that tunes the accuracy of the approximation[2].

The next result studies the basic properties of Algorithm 3.

Proposition 3.5.1. ***(Properties of Algorithm 3).*** *Let A and B be matrices of equal size, $\epsilon > 0$, and $Z = \text{approx-null}(A, B, \epsilon)$. Then,*

(i) Algorithm 3 terminates in finite iterations;

(ii) Z is either $\emptyset$ or has full column rank;

(iii) if $Z \neq \emptyset$, let $Z = [(Z^A)^T, (Z^B)^T]^T$ with Z^A, Z^B of equal size. Then Z^A and Z^B have full column rank.

Proof. (i) We prove it by contradiction, i.e., suppose the algorithm does not terminate in finite iterations. Let Z_i be the internal matrix in Step 8 at iteration i. Since by construction

[2]In Algorithm 3, A and B have equal size and both have full column rank.

Algorithm 3 approx-null(A, B, ϵ)

```
    Inputs: $A, B$ matrices, $\epsilon > 0$   Output: $Z$                    $\triangleright$ $A, B$ have the same size
    Procedure: $Z \leftarrow$ approx-null(A, B, $\epsilon$)
    $\triangledown$ Singular value decomposition of $[A, B]$
 1: $\{U, S, V\} \leftarrow \mathrm{svd}([A, B])$                              $\triangleright$ $USV^T = [A, B]$
 2: $m \leftarrow \sharp\mathrm{cols}(V)$                                      $\triangleright$ # of columns of $V$
 3: $k_{\min} \leftarrow \left\{ \min_k \text{ s. t. } \left( \frac{\sum_{i=k}^{m} S_{i,i}^2}{\|S\|_F^2} \leq \epsilon^2 \wedge k > \sharp\mathrm{cols}(A) \right) \right\}$
    $\triangledown$ Choosing the right singular vectors corresponding to small singular values as the
    approximated null space
 4: if $k_{\min} = \emptyset$ then
 5:     return $\emptyset$
 6:     break
 7: else
 8:     $Z \leftarrow (V_{k_{\min}:m}^T)^T$
 9: end if
    $\triangledown$ Make sure Assumption 3.1.3 holds for the output
10: while 1 do
11:     $\begin{bmatrix} Z^A \\ Z^B \end{bmatrix} \leftarrow Z$                 $\triangleright$ $\sharp\mathrm{rows}(Z^A) = \sharp\mathrm{rows}(Z^B)$
12:     if $\mathrm{rank}(Z^A) = \mathrm{rank}(Z^B) = \sharp\mathrm{cols}(Z)$ then
13:         return $Z$                                    $\triangleright$ Basis for approximated null space
14:         break
15:     end if
        $\triangledown$ Reducing the space
16:     if $\sharp\mathrm{cols}(Z) = 1$ then
17:         return $\emptyset$
18:         break
19:     else
20:         $Z \leftarrow (Z_{2:\sharp\mathrm{cols}(Z)}^T)^T$              $\triangleright$ Removing the 1st column
21:     end if
22: end while
```

$k_{\min} > \sharp\text{cols}(A)$ and $m = \sharp\text{cols}(V) = \sharp\text{cols}(A) + \sharp\text{cols}(B)$ (cf. Step 2), we deduce

$$\sharp\text{cols}(Z_1) = m - k_{\min} \leq \sharp\text{cols}(B). \tag{3.48}$$

Moreover, since we assumed the algorithm never terminates, it executes Step 20 at each iteration and consequently, $\sharp\text{cols}(Z_{i+1}) = \sharp\text{cols}(Z_i) - 1$ for $i \in \mathbb{N}$. As a result, one can use (3.48) to write

$$\sharp\text{cols}(Z_j) = \sharp\text{cols}(Z_1) - j + 1 \leq \sharp\text{cols}(B) - j + 1,$$

which leads to $\sharp\text{cols}(Z_j) < 0$ for $j > \sharp\text{cols}(B) + 1$, contradicting $\sharp\text{cols}(Z_j) \geq 0$.

(ii) There are three ways for Algorithm 3 to terminate: either Steps 13-14, Steps 5-6, or Steps 17-18. The latter two cases imply $Z = \emptyset$. In the other case, since the columns of Z are selected from the right singular vectors of $[A, B]$, they are nonzero and mutually orthogonal. Consequently, Z has full column rank.

(iii) Since $Z \neq \emptyset$, the algorithm executes Steps 13-14 upon termination. Hence, the condition in Step 12 holds, and consequently Z^A and Z^B have full column rank. □

We next characterize the quality of Algorithm 3's output.

Proposition 3.5.2. ***(Quality of Low-Rank Approximation of*** $[A, B]$ ***Constructed with Output of Algorithm 3).*** *Let $\epsilon > 0$, A and B full column rank matrices with equal size, and assume $Z = \text{null-approx}(A, B, \epsilon) \neq \emptyset$. Denote $W = [A, B]$ and let $W = USV^T$ be its singular value decomposition. Let $\bar{S}$ be defined by setting in S the entries $\bar{S}_{i,i} = 0$*

for $i \in \{\sharp\mathrm{cols}(V) - \sharp\mathrm{cols}(Z) + 1, \ldots, \sharp\mathrm{cols}(V)\}$*. Define* $\bar{W} = U\bar{S}V^T$ *and express it as the concatenation* $\bar{W} = [\bar{A}, \bar{B}]$*, where* $\bar{A}$ *and* $\bar{B}$ *have the same size. Then,*

(i) $\|W - \bar{W}\|_F \leq \epsilon \|W\|_F$*;*

(ii) the columns of Z *form a basis for the null space of* $\bar{W}$*;*

(iii) $\bar{A}Z^A = -\bar{B}Z^B$*, where* $Z = [(Z^A)^T, (Z^B)^T]^T$ *and* Z^A, Z^B *have the same size.*

Proof. (i) By construction we have $Z = V^T_{(\sharp\mathrm{cols}(V) - \sharp\mathrm{cols}(Z)+1):\sharp\mathrm{cols}(V)}$, i.e., the columns of Z are the last $\sharp\mathrm{cols}(Z)$ columns of V, corresponding to the smallest singular values of W. Moreover, based on Step 3 of the algorithm, the fact that the singular values are ordered in a decreasing manner in S, and noting that $k_{\min} \leq \sharp\mathrm{cols}(V) - \sharp\mathrm{cols}(Z) + 1$, one can write

$$\sum_{i=\sharp\mathrm{cols}(V)-\sharp\mathrm{cols}(Z)+1}^{\sharp\mathrm{cols}(V)} S_{i,i}^2 \leq \epsilon^2 \sum_{i=1}^{\sharp\mathrm{cols}(V)} S_{i,i}^2 = \epsilon^2 \|W\|_F^2.$$

The proof concludes by noting that the left hand side term in the previous equation is equal to $\|W - \bar{W}\|_F^2$.

(ii) The proof directly follows from the fact that $\bar{W} = U\bar{S}V^T$ is the singular value decomposition of $\bar{W}$ and the columns of Z are the right singular vectors corresponding to zero singular values of $\bar{W}$.

(iii) Based on (ii), $\bar{W}Z = \mathbf{0}$. Hence, $\bar{A}Z^A = -\bar{B}Z^B$. □

We formally define the *Approximated-SSD* algorithm (cf. Algorithm 4) as the modification of SSD that replaces Step 4 of Algorithm 1 by

$$\begin{bmatrix} Z_i^A \\ \\ Z_i^B \end{bmatrix} \leftarrow \text{approx-null}(A_i, B_i, \epsilon).$$

Since all other steps of Approximated-SSD are identical to SSD, we omit presenting it for space reasons.

Algorithm 4 Approximated Symmetric Subspace Decomposition

```
    **Inputs:** $D(X), D(Y) \in \mathbb{R}^{N \times N_d}, \epsilon > 0$    **Output:** $C_{\text{SSD}}^{\text{aprx}}$
    **Procedure:** $C_{\text{SSD}}^{\text{aprx}} \leftarrow$ Approximated-SSD$\big(D(X), D(Y), \epsilon\big)$
 1: **Initialization**
 2: $i \leftarrow 1$, $A_1 \leftarrow D(X)$, $B_1 \leftarrow D(Y)$, $C_{\text{SSD}}^{\text{aprx}} \leftarrow I_{N_d}$
 3: **while** 1 **do**
 4:     $\begin{bmatrix} Z_i^A \\ Z_i^B \end{bmatrix} \leftarrow$ approx-null$(A_i, B_i, \epsilon)$.        ▷ Calling Algorithm 3
 5:     **if** approx-null$(A_i, B_i, \epsilon) = \emptyset$ **then**
 6:         **return** 0                                   ▷ The basis does not exist
 7:         **break**
 8:     **end if**
 9:     **if** $\sharp$rows$(Z_i^A) \leq \sharp$cols$(Z_i^A)$ **then**
10:         **return** $C_{\text{SSD}}^{\text{aprx}}$                 ▷ The procedure is complete
11:         **break**
12:     **end if**
13:     $C_{\text{SSD}}^{\text{aprx}} \leftarrow C_{\text{SSD}}^{\text{aprx}} Z_i^A$                    ▷ Reducing the subspace
14:     $A_{i+1} \leftarrow A_i Z_i^A$, $B_{i+1} \leftarrow B_i Z_i^A$, $i \leftarrow i + 1$
15: **end while**
```

Proposition 3.5.1 completely preserves the logical structure for the proof of Theorem 3.3.1(i) and, as a result, we deduce that the Approximated-SSD algorithm terminates in at most N_d iterations. Moreover, the $C_{\text{SSD}}^{\text{aprx}}$ matrix is zero or has full column rank, since the second part of the proof for Theorem 3.3.1(ii) also holds for Approximated-SSD. If $C_{\text{SSD}}^{\text{aprx}} \neq 0$, one can define the reduced dictionary with $\tilde{N}_d = \sharp\text{cols}(C_{\text{SSD}}^{\text{aprx}})$ elements as

$$\tilde{D}_{\text{aprx}}(\cdot) := D(\cdot)C_{\text{SSD}}^{\text{aprx}}. \tag{3.49}$$

We propose calculating the linear prediction matrix $K_{\text{SSD}}^{\text{aprx}}$ by solving the following total least squares (TLS) problem (see e.g. [MH07] for more information on TLS)

$$\begin{aligned} &\underset{K,\Delta_1,\Delta_2}{\text{minimize}} && \|[\Delta_1, \Delta_2]\|_F && \text{(3.50a)} \\ &\text{subject to} && \tilde{D}_{\text{aprx}}(Y) + \Delta_2 = (\tilde{D}_{\text{aprx}}(X) + \Delta_1)K. && \text{(3.50b)} \end{aligned}$$

Even though TLS problems do not always have a solution, the next result shows that (3.50) does. We also provide its closed-form solution and a bound on the accuracy of the prediction on the available data based on the parameter ϵ.

Theorem 3.5.3. ***(Solution and Prediction Accuracy of*** (3.50)***).*** *Let* $[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)] = USV^T$ *be the singular value decomposition of* $[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$*. Let* $\bar{S}$ *be defined by setting in* S *the entries* $\bar{S}_{i,i} = 0$ *for* $i \in \{\tilde{N}_d + 1, \ldots, 2\tilde{N}_d\}$*. Let* $U\bar{S}V^T = [\bar{A}, \bar{B}]$*, with* $\bar{A}$*,* $\bar{B}$ *of the same size. Define*

$$K_{\text{SSD}}^{\text{aprx}} = \bar{A}^{\dagger}\bar{B}, \tag{3.51a}$$

$$[\Delta_1^*, \Delta_2^*] = [\bar{A}, \bar{B}] - [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]. \tag{3.51b}$$

Then, $K_{\text{SSD}}^{\text{aprx}}$*,* Δ_1^**,* Δ_2^* *are the global solution of* (3.50) *and*

$$\|[\Delta_1^*, \Delta_2^*]\|_F \leq \epsilon \|[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]\|_F. \tag{3.52}$$

Proof. One can rewrite (3.50b) as

$$\big([\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)] + [\Delta_1, \Delta_2]\big) \begin{bmatrix} K \\ -I_{\tilde{N}_d} \end{bmatrix} = 0,$$

which implies that $\text{rank}([\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)] + [\Delta_1, \Delta_2]) \leq \tilde{N}_d$. Using Eckart-Young theorem [EY36], one deduces that $[\bar{A}, \bar{B}]$ is the closest matrix (in Frobenius norm) to $[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$ of rank smaller than or equal to $\tilde{N}_d$. In other words, Δ_1^* and Δ_2^* in (3.51b) minimize the cost function in (3.50a). Next, we need to show that they also satisfy (3.50b) with $K_{\text{SSD}}^{\text{aprx}}$ defined in (3.51a).

Let t be the termination iteration of the Approximated-SSD algorithm. Since $C_{\text{SSD}}^{\text{aprx}} \neq 0$, the algorithm executes Step 10. Therefore, the condition in Step 9 holds and $\sharp\text{rows}(Z_t^A) \leq \sharp\text{cols}(Z_t^A)$, where $[(Z_t^A)^T, (Z_t^B)^T]^T = \text{approx-null}(A_t, B_t, \epsilon)$. In addition, based on Proposition 3.5.1(iii), Z_t^A and Z_t^B are nonsingular square matrices. Noting that by definition in the Approximated-SSD algorithm, $A_t = D(X)C_{\text{SSD}}^{\text{aprx}} = \tilde{D}_{\text{aprx}}(X)$ and $B_t = D(Y)C_{\text{SSD}}^{\text{aprx}} = \tilde{D}_{\text{aprx}}(Y)$, one can use Proposition 3.5.2(iii) with $W = [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$ and $\bar{W} = [\bar{A}, \bar{B}]$ to write $\bar{A}Z_t^A = -\bar{B}Z_t^B$. Since Z_t^A and Z_t^B are nonsingular square matrices, the previous equation leads to $\mathcal{R}(\bar{A}) = \mathcal{R}(\bar{B})$ and $\bar{A}K_{\text{SSD}}^{\text{aprx}} = \bar{B}$, where $K_{\text{SSD}}^{\text{aprx}}$ is defined in (3.51a). As a result, $\Delta_1^*, \Delta_2^*, K_{\text{SSD}}^{\text{aprx}}$ satisfy the constraint (3.50b). Finally, the accuracy bound defined in (3.52) follows from Proposition 3.5.2(i) with $W = [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$ and $\bar{W} = [\bar{A}, \bar{B}]$. $\square$

Note that, unlike in the exact case (cf. Theorem 3.3.8), Theorem 3.5.3 does not provide an out-of-sample bound on prediction accuracy. According to this result, a small perturbation $[\Delta_1^*, \Delta_2^*]$ to the matrix $[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$ allows us to describe the evolution of the dictionary matrices linearly through $K_{\text{SSD}}^{\text{aprx}}$. Moreover, the Frobenius norm of the perturbation is upper bounded by $\epsilon \| [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)] \|_F$, which implies that a smaller ϵ leads to better accuracy on the observed samples.

3.6 Simulation Results

We illustrate the efficacy of the proposed methods in two examples.[3]

Example 3.6.1. ***(Unstable Discrete-time Polynomial System).*** Consider the nonlinear system

$$\begin{aligned} x_1^+ &= 1.1\, x_1 \\ x_2^+ &= 1.2\, x_2 + 0.1\, x_1^2 + 0.1, \end{aligned} \tag{3.53}$$

with state $x^T = [x_1, x_2]^T$. System (3.53) is actually an unstable Polyflow [JT19] which has a state-inclusive finite-dimensional Koopman-invariant subspace comprised of polynomials.

[3]We have chosen on purpose low-dimensional examples to be able to fully detail the identified Koopman eigenvalues and associated subspaces. However, it is worth pointing out that the results presented here are applicable without any restriction on the type of dynamical system, its dimension, or the sparsity of the model in the dictionary.

We use the dictionary $D(x) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1x_2^2, x_1^2x_2, x_2^3]$ with $N_d = 10$. Moreover, we gather 2×10^4 data snapshots uniformly sampled from $[-2, 2] \times [-2, 2]$. We use the SSD and SSSD strategies to identify the maximal Koopman-invariant subspaces in $\operatorname{span}(D)$. In the SSSD method, we use the first 10 data snapshots as signature snapshots and feed the rest of the data to the algorithm according to the order they appear in the data set. We use the strategy explained in Remark 3.3.4 with $\epsilon = 10^{-12}$ to overcome error due to the use of finite-precision machines. Both methods find bases for the 6-dimensional subspace spanned by $\{1, x_1, x_2, x_1x_2, x_1^2, x_1^3\}$, which is the maximal Koopman-invariant subspace in $\operatorname{span}(D)$. The SSSD method, however, performs the calculations 96% faster than SSD. One can find K_{SSD} by applying EDMD on either of the identified dictionaries according to (3.21). Moreover, based on Theorems 3.3.5 and 3.3.7, we use the eigendecomposition of K_{SSD} to find all the Koopman eigenfunctions associated with the system (3.53) in $\operatorname{span}(D)$. Table 3.1 shows the identified eigenfunctions. One can use direct calculation to verify that the identified functions are the Koopman eigenfunctions associated with system (3.53). Note that since x_1 and x_2 are both in the span of the identified Koopman-invariant subspace, one can fully characterize the behavior of the system using the eigenfunctions and (2.8) linearly or directly using the identified dictionary and K_{SSD}.

Next, we evaluate the effectiveness of the original dictionary D and the dictionary D_{SSD} identified by SSD (equivalently, by SSSD) for long-term prediction. To do this, we consider error functions defined as follows. Given an arbitrary dictionary $\mathcal{D}$, consider its associated matrix $K = \text{EDMD}(\mathcal{D}, X, Y)$. For a trajectory $\{x(k)\}_{k=0}^{M}$ of (3.53) with length

Table 3.1: Identified eigenfunctions and eigenvalues of the Koopman operator associated with system (3.53).

Eigenfunction	**Eigenvalue**
$\phi_1(x) = 1$	$\lambda_1 = 1$
$\phi_2(x) = x_1$	$\lambda_2 = 1.1$
$\phi_3(x) = x_1^2$	$\lambda_3 = 1.21$
$\phi_4(x) = 20\,x_1^2 - 2\,x_2 - 1$	$\lambda_4 = 1.2$
$\phi_5(x) = x_1^3$	$\lambda_5 = 1.331$
$\phi_6(x) = 20\,x_1^3 - 2\,x_1 x_2 - x_1$	$\lambda_6 = 1.32$

M and initial condition x_0, let

$$E_{\text{relative}}(k) = \frac{\left\|\mathcal{D}(x(k)) - \mathcal{D}(x_0)K^k\right\|_2}{\|\mathcal{D}(x(k))\|_2} \times 100, \tag{3.54a}$$

$$E_{\text{angle}}(k) = \angle\big(\mathcal{D}(x(k)), \mathcal{D}(x_0)K^k\big), \tag{3.54b}$$

where $\mathcal{D}(x_0)K^k$ is the predicted dictionary vector at time k calculated using the dictionary $\mathcal{D}$. E_{relative} corresponds to the relative error in magnitude between the predicted and exact dictionary vectors and E_{angle} corresponds to the error in the angle of the vectors.

We compute the errors associated to the original dictionary D, denoted $E_{\text{relative}}^{\text{Orig}}$ and $E_{\text{angle}}^{\text{Orig}}$, and the errors associated to the SSD dictionary D_{SSD}, denoted $E_{\text{relative}}^{\text{SSD}}$ and $E_{\text{angle}}^{\text{SSD}}$. Figure 3.1 illustrates these errors along a trajectory starting from a random initial condition in $[-2,2] \times [-2,2]$ for 20 time steps. The plot shows the importance of the dictionary selection when performing EDMD. Unlike the prediction on span(D), the prediction on the SSD subspace span(D_{SSD}) matches the behavior of the system exactly. This is a direct consequence of the fact that span(D_{SSD}) is a Koopman-invariant subspace, on which EDMD fully captures the behavior of the operator through K_{SSD}. It is worth mentioning that based on Proposition 3.2.2, EDMD with dictionary D also predicts the functions in $\mathrm{span}(D_{\mathrm{SSD}})$ exactly. However, its prediction for functions outside of $\mathrm{span}(D_{\mathrm{SSD}})$ leads to large errors.

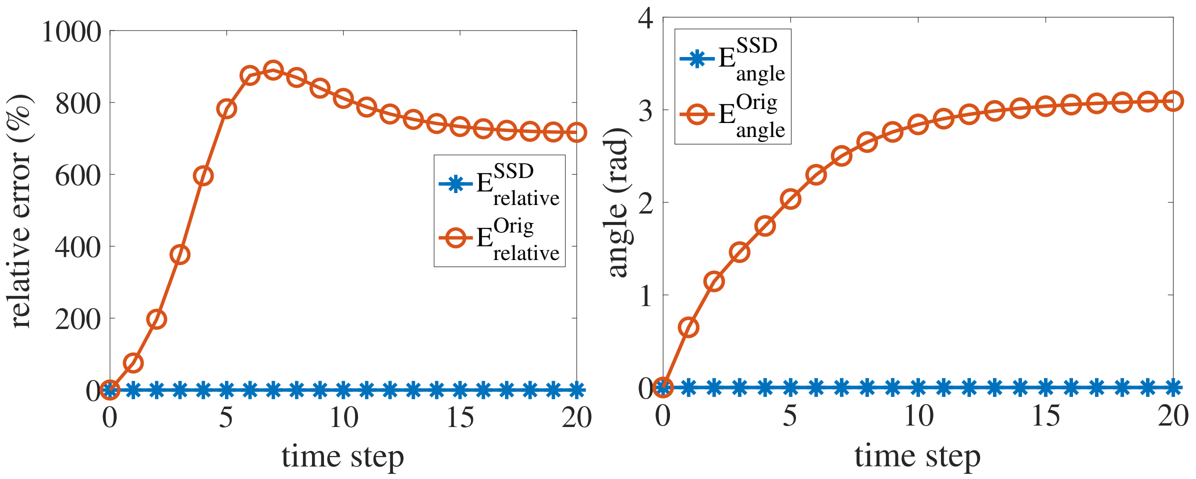

Figure 3.1: Relative (left) and angle (right) prediction errors on the original and SSD subspaces for system (3.53) on a trajectory of length $M = 20$.

Example 3.6.2. ***(Duffing Equation).*** Here, we investigate the efficacy of the proposed methods in approximating Koopman eigenfunctions and invariant subspaces. Consider the Duffing equation [WKR15, Section 4.2]

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -0.5x_2 + x_1 - x_1^3, \end{aligned} \tag{3.55}$$

with state $x^T = [x_1, x_2]^T$. The system has one unstable equilibrium at the origin and two locally stable equilibria at $[\pm 1, 0]^T$. We consider the discretized version of (3.55) with timestep $\Delta t = 0.01s$ and gather $N = 5000$ data snapshots uniformly sampled from $\mathcal{M} = [-2, 2] \times [-2, 2]$. Moreover, we use the dictionary D comprised of all $N_d = 36$ monomials up to degree 7 in the form of $\prod_{i=1}^{7} y_i$, where $y_i \in \{1, x_1, x_2\}$. The maximal Koopman-invariant subspace in the span of the dictionary is one dimensional, spanned by the trivial eigenfunction $\phi \equiv 1$. Hence, applying the SSD and SSSD algorithms would result in a trivial solution. Instead, we apply the Approximated-SSD algorithm on the available dictionary snapshots with the accuracy parameter $\epsilon = 10^{-3}$. The outcome is the dictionary $\tilde{D}_{\text{aprx}}$ with $\tilde{N}_d = 15$ elements. We calculate the linear prediction matrix $K_{\text{SSD}}^{\text{aprx}}$ using Theorem 3.5.3. The norm of the perturbation $\|[\Delta_1^*, \Delta_2^*]\|_F$ satisfies

$$\|[\Delta_1^*, \Delta_2^*]\|_F \approx 9.6 \times 10^{-4} \|[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]\|_F,$$

agreeing with the upper bound provided in Theorem 3.5.3. We approximate the eigenfunctions of the Koopman operator using the eigendecomposition of $K_{\text{SSD}}^{\text{aprx}}$. For space reasons, we only illustrate the leading nontrivial approximated Koopman eigenfunctions with eigenvalue closest to the unit circle. Figure 3.2(left) shows the real-valued approximated eigenfunction corresponding to the eigenvalue $\lambda = 0.9919$. Despite being an approximation, the eigenfunction captures the behavior of the vector field accurately and correctly identifies the attractiveness of the two locally stable equilibria. Given that $|\lambda| < 1$, Figure 3.2(left) predicts that the trajectories eventually converge to one of the stable equilibria. Figure 3.2(right) shows the absolute value of the approximated Koopman eigenfunctions corresponding to the eigenvalues $\lambda = 0.9989 \pm 0.0037j$. Similarly to the other plot, it captures information about the shape of the vector field such as the attractive equilibria and their regions of attraction.

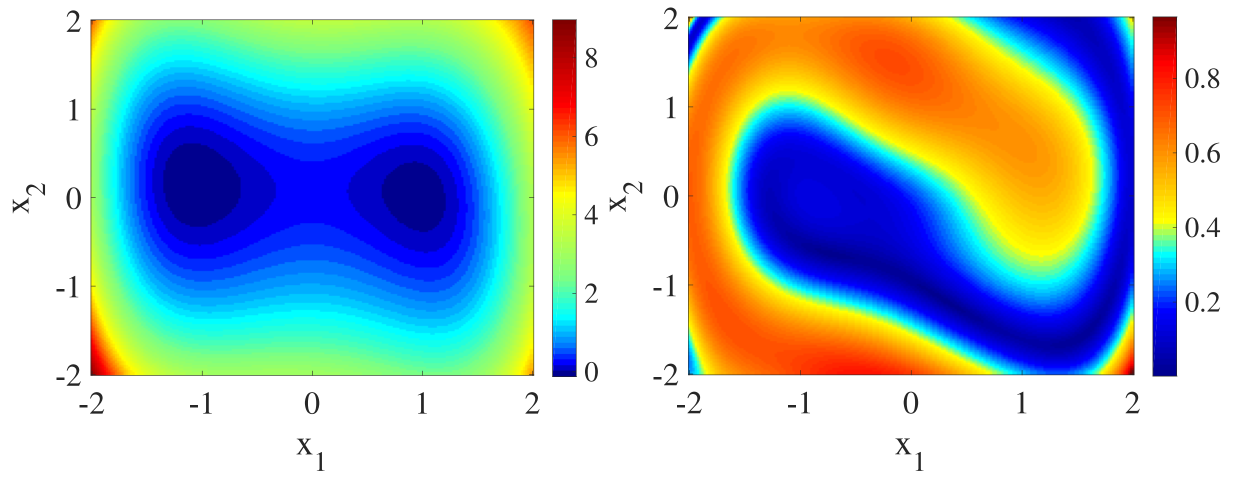

Figure 3.2: The eigenfunction corresponding to the eigenvalue $\lambda = 0.9919$ (left) and the absolute value of the eigenfunctions corresponding to the eigenvalues $\lambda = 0.9989 \pm 0.0037j$ (right) for the Koopman operator associated with (3.55), as identified by the Approximated-SSD algorithm.

We use the relative and angle errors defined in (3.54) to compare the prediction accuracy of the original dictionary D and the Approximated-SSD dictionary $\tilde{D}_{\text{aprx}}$ (for which we use $K_{\text{SSD}}^{\text{aprx}}$). Figure 3.3 illustrates the relative and angle errors along a trajectory starting from a random initial condition in $[-2, 2] \times [-2, 2]$ for 30 timesteps. The plot shows the superiority of EDMD over the subspace identified by Approximated-SSD in long-term prediction of dynamical behavior.

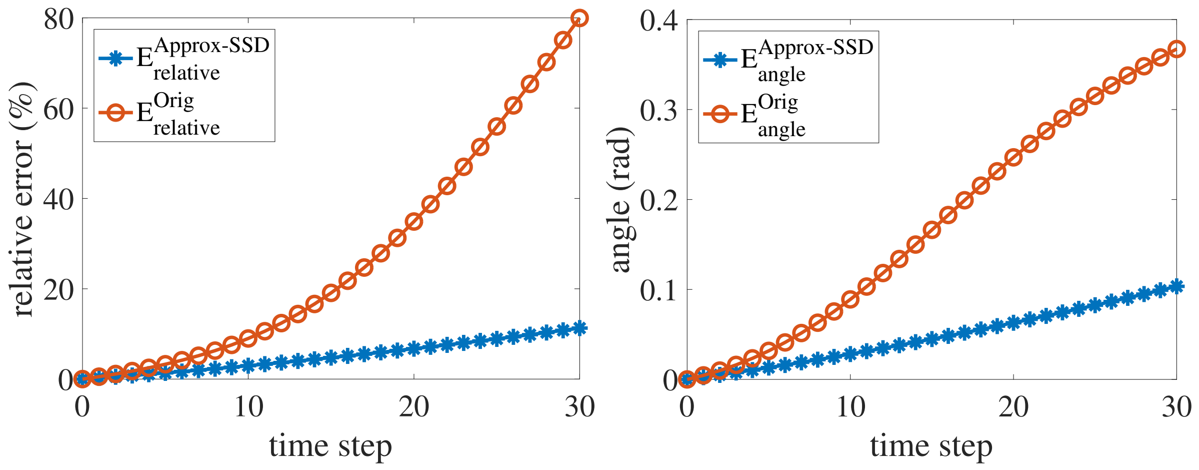

Figure 3.3: Relative (left) and angle (right) prediction errors on Approximated-SSD and original subspaces for system (3.55) on a trajectory of length $M = 30$.

3.7 Chapter Appendix

Here, we provide several linear algebraic results that we have used throughout the chapter.

Lemma 3.7.1. ***(Intersection of Linear Spaces).*** *Let $A, B \in \mathbb{R}^{m\times n}$ be matrices with full column rank. Suppose that the columns of $Z = [(Z^A)^T, (Z^B)^T]^T \in \mathbb{R}^{2n\times l}$ form a basis for the null space of $[A, B]$, where $Z^A, Z^B \in \mathbb{R}^{n\times l}$. Then,*

(i) $\mathcal{R}(AZ^A) = \mathcal{R}(A) \cap \mathcal{R}(B)$;

(ii) Z^A and Z^B have full column rank.

Proof. (i) First, note that $\mathcal{R}(AZ^A) \subseteq \mathcal{R}(A)$. Moreover, by hypothesis, $[A, B]Z = 0$, which leads to $\mathcal{R}(AZ^A) = \mathcal{R}(BZ^B) \subseteq \mathcal{R}(B)$. Consequently, $\mathcal{R}(AZ^A) \subseteq \mathcal{R}(A) \cap \mathcal{R}(B)$. Now, suppose that $v \in \mathcal{R}(A) \cap \mathcal{R}(B)$. By definition, there exist vectors $w_1, w_2 \in \mathbb{R}^n$ such that

$v = Aw_1 = Bw_2$. Then,

$$\begin{bmatrix} A, B \end{bmatrix} \begin{bmatrix} w_1 \\ -w_2 \end{bmatrix} = 0,$$

which means that $[w_1^T, -w_2^T]^T \in \mathcal{R}(Z)$ and there exists $r \in \mathbb{R}^l$ such that $w_1 = Z^A r$ and $w_2 = -Z^B r$. Therefore, $v = Aw_1 = AZ^A r \in \mathcal{R}(AZ^A)$. Consequently, $\mathcal{R}(A) \cap \mathcal{R}(B) \subseteq \mathcal{R}(AZ^A)$ and the claim follows.

(ii) We prove this part using contradiction. Suppose that there exists $v \neq 0$ such that $Z^A v = 0$. Also, since $[A, B]Z = 0$, one can conclude that $AZ^A v = -BZ^B v = 0$. Since B has full column rank, we have $Z^B v = 0$. Hence $Zv = 0$, which is a contradiction since the columns of Z are linearly independent. A similar reasoning shows that Z^B has full column rank. □

Lemma 3.7.2. *Let A, C, D be matrices of appropriate sizes, with A having full column rank. Then $\mathcal{R}(AC) \subseteq \mathcal{R}(AD)$ if and only if $\mathcal{R}(C) \subseteq \mathcal{R}(D)$.*

Proof. ($\Rightarrow$): Suppose that $v \in \mathcal{R}(C)$, and hence there exists w such that $v = Cw$. Therefore, $Av = ACw \in \mathcal{R}(AC)$. Since $\mathcal{R}(AC) \subseteq \mathcal{R}(AD)$, one can deduce that there exist r such that $ACw = ADr$, and we get $A(v - Dr) = 0$. This leads to $v = Dr \in \mathcal{R}(D)$ since A has full column rank, and hence $\mathcal{R}(C) \subseteq \mathcal{R}(D)$.

($\Leftarrow$): Suppose that $v \in \mathcal{R}(AC)$, and hence $v = ACw$ for some w. Since $\mathcal{R}(C) \subseteq \mathcal{R}(D)$, there exists r such that $Cw = Dr$. As a result, $v = ACw = ADr$ which leads to the conclusion that $v \in \mathcal{R}(AD)$ and the claim follows. □

Lemma 3.7.3. *Let $A_1, B_1 \in \mathbb{R}^{m\times n}$, $A_2, B_2 \in \mathbb{R}^{l\times n}$, and $C \in \mathbb{R}^{n\times k}$ with A_1, B_1, C having full column rank. If*

$$\mathcal{R}\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right) = \mathcal{R}\left(\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\right), \tag{3.56a}$$

$$\mathcal{R}(A_1C) = \mathcal{R}(B_1C), \tag{3.56b}$$

then

$$\mathcal{R}\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} C\right) = \mathcal{R}\left(\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} C\right).$$

Proof. Based on (3.56a), one can deduce that there exists a nonsingular square matrix K such that

$$A_1 = B_1K, \tag{3.57a}$$

$$A_2 = B_2K. \tag{3.57b}$$

Multiplying both sides of (3.57a) by C gives

$$A_1C = B_1KC. \tag{3.58}$$

Moreover, based on (3.56b), one can deduce that there exists a nonsingular square matrix

K^*

$$A_1 C = B_1 C K^*. \tag{3.59}$$

By subtracting (3.59) from (3.58) and considering the fact that B_1 has full column rank, one can deduce that

$$C K^* = K C. \tag{3.60}$$

Now, multiplying both sides of (3.57b) from the right by C and using (3.60), one can write $A_2 C = B_2 C K^*$, which in conjunction with (3.59) leads to

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} C = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} C K^*,$$

completing the proof. □

Lemma 3.7.4. *Let A, $\{C_i\}_{i=1}^{\infty}$, and $\hat{C}$ be matrices of appropriate sizes. Assume that A has full column rank and $\mathcal{R}(\hat{C}) = \bigcap_{i=1}^{\infty} \mathcal{R}(C_i)$. Then $\mathcal{R}(A\hat{C}) = \bigcap_{i=1}^{\infty} \mathcal{R}(AC_i)$.*

Proof. First, we prove that $\mathcal{R}(A\hat{C}) \subseteq \bigcap_{i=1}^{\infty} \mathcal{R}(AC_i)$. Let $v \in \mathcal{R}(A\hat{C})$, then there exists a vector w such that $v = A\hat{C}w = Ar$, with $r = \hat{C}w$. Note that $r \in \mathcal{R}(\hat{C})$ and consequently $r \in \mathcal{R}(C_i)$ for all $i \in \mathbb{N}$. Hence, for every $i \in \mathbb{N}$ there exists z_i such that $r = C_i z_i$. Based on the definition of v, we have $v = Ar = AC_i z_i$ for every $i \in \mathbb{N}$. As a result, $v \in \bigcap_{i=1}^{\infty} \mathcal{R}(AC_i)$ which concludes the proof of this inclusion.

Now, we prove that $\bigcap_{i=1}^{\infty} \mathcal{R}(AC_i) \subseteq \mathcal{R}(A\hat{C})$. Let $v \in \bigcap_{i=1}^{\infty} \mathcal{R}(AC_i)$. Then for every $i \in \mathbb{N}$, there exists w_i such that $v = AC_i w_i$. Moreover, $v \in \mathcal{R}(A)$ and since A has full column rank, there exists a unique r such that $v = Ar$. Therefore, for all $i \in \mathbb{N}$, we have $r = C_i w_i$. Thus, $r \in \bigcap_{i=1}^{\infty} \mathcal{R}(C_i)$ and consequently, $r \in \mathcal{R}(\hat{C})$. Since $v = Ar$, we have $v \in \mathcal{R}(A\hat{C})$, concluding the proof. $\square$

Acknowledgements

This chapter, in part, is a reprint of the material [HC22a] as it appears in 'Learning Koopman Eigenfunctions and Invariant Subspaces from Data: Symmetric Subspace Decomposition' by M. Haseli and J. Cortés, in IEEE Transactions on Automatic Control 2022. The dissertation author was the primary investigator and author of this paper. The work in this chapter was partially supported by ONR Award N00014-18-1-2828.

Chapter 4

Parallel Identification of Koopman Eigenfunctions and Invariant Subspaces

Given the recent advances in parallel and cloud computing structures, designing methods that can utilize such hardware is of utmost importance. This chapter aims at providing parallel computing methods to find the maximal Koopman-invariant subspace in the span of an arbitrary dictionary of functions using data gather from trajectories of an unknown nonlinear system. In other words, this chapter presents a parallel counterpart to the Symmetric Subspace Decomposition (SSD) algorithm, cf. Algorithm 1, presented in Chapter 3. We consider a network of processors each aware of a common dictionary of functions and equipped with a local set of data snapshots. The processors repeatedly interact over a directed communication graph. Each processor receives its neighbors' estimates of the invariant dictionary and refines its estimate by applying SSD with its local data on the intersection of the subspaces spanned by its own dictionary and the neighbors' dictionaries. We identify conditions on the network topology to ensure the algorithm identifies the maximal Koopman-invariant subspace in the span of the original dictionary, characterize its time, computational, and communication complexity, and establish its robustness against communication failures.

4.1 Problem Statement

Our aim here is to use parallel computation to find Koopman eigenfunctions and invariant subspaces. In other words, we aim to use parallel processing to speed up the methods provided in Chapter 3[1]. Consider the following discrete-time system defined over the state space $\mathcal{M} \subseteq \mathbb{R}^n$

$$x^+ = T(x), \tag{4.1}$$

and let $\mathcal{K}$ be its associated Koopman operator defined on the linear space of functions $\mathcal{F}$ comprised of functions mapping elements in $\mathcal{M}$ to $\mathcal{M}$. Given an arbitrary finite-dimensional functional space $\mathcal{P} \subseteq \mathcal{F}$, our goal is to design efficient data-driven algorithms that are able to identify the maximal Koopman-invariant subspace in $\mathcal{P}$ associated with a dynamical system whose explicit model is unknown. To do so, we rely on data collected about the dynamical behavior of the system and aim to ensure that the proposed methods are compatible with parallel processing hardware.

[1]Some problem elements discussed here are similar to the elements used in Section 3.1 and are repeated for completeness.

We start by defining the concepts of data snapshots and dictionary. We represent by $X, Y \in \mathbb{R}^{N \times n}$ matrices comprised of N data snapshots acquired from system (4.1) such that x_i^T and y_i^T, the ith rows of X and Y, satisfy

$$y_i = T(x_i), \quad \forall i \in \{1, \ldots, N\}.$$

Moreover, we define a dictionary $D : \mathcal{M} \to \mathbb{R}^{1 \times N_d}$

$$D(\cdot) = [d_1(\cdot), \ldots, d_{N_d}(\cdot)],$$

comprised of N_d real-valued functions $d_1, \ldots, d_{N_d}$ defined on $\mathcal{M}$ with $\operatorname{span}(D) = \mathcal{P}$. Note that, one can completely characterize any dictionary $\tilde{D}$ with elements in $\operatorname{span}(D)$ by a matrix C with N_d rows as $\tilde{D}(\cdot) = D(\cdot)C$. The effect of the dictionary on data matrices is

$$D(X) = [D(x_1)^T, \ldots, D(x_N)^T]^T \in \mathbb{R}^{N \times N_d}.$$

Throughout this chapter, we operate under the following standard assumption on dictionary matrices.

Assumption 4.1.1. ***(Full Column Rank Dictionary Matrices).*** *The matrices $D(X)$ and $D(Y)$ have full column rank.* $\square$

Assumption 4.1.1 requires the dictionary functions to be linearly independent and the data snapshots to be diverse enough to capture the important features of the dynamics.

We consider a group of M processors or agents that communicate according to a digraph G. Every agent is aware of the dictionary. We assume the data snapshots X, Y are distributed among the processors, forming the dictionary snapshots $D(X_i), D(Y_i)$ for each agent $i \in \{1, \ldots, M\}$ such that

$$\bigcup_{i=1}^{M} \text{rows}([D(X_i), D(Y_i)]) = \text{rows}([D(X), D(Y)]).$$

Local data snapshots might be available at the agent as a result of distributed acquisition or because the global data has been partitioned among the processors. The latter can be done in different ways, including partitioning in a way that minimizes the maximum time taken for each agent to finish one iteration of calculation. For homogeneous processors, this can be done by evenly distributing the the data among the agents.

The processors share a small portion of dictionary snapshots, $D(X_s)$ and $D(Y_s)$ with full column rank, called the *signature dictionary matrices* (note that one can build signature dictionary snapshots with no more than $2N_d$ rows according to Assumption 4.1.1). Hence, we have

$$\text{rows}([D(X_s), D(Y_s)]) \subseteq \text{rows}([D(X_i), D(Y_i)]),$$

for $i \in \{1, \ldots, M\}$. Interestingly, for our forthcoming results, the processors do not need to be aware of which data points correspond to the signature snapshots.

Problem Statement: Given a group of M processors communicating according to a network, the dictionary $D(\cdot) = [d_1(\cdot), \ldots, d_{N_d}(\cdot)]$, and the data snapshots $X, Y \in \mathbb{R}^{N \times n}$, we aim to design a parallel algorithm that is able to identify the dictionary $\tilde{D}$ spanning the maximal Koopman-invariant subspace in $\text{span}(D)$. The processors must only rely on the their local data and communication with their neighbors to achieve this goal.

In general, the action of Koopman operator on the functional space $\mathcal{P} = \text{span}(D)$ is not invariant, meaning that one needs to project back to $\mathcal{P}$ the image under $\mathcal{K}$ of functions in $\mathcal{P}$, thereby introducing error. Instead, the image of functions belonging to the identified invariant subspace under the Koopman operator remains in the subspace, avoiding the need for the projection and therefore eliminating the source of approximation error. We note that efficient solutions to this problem could be employed in the construction of better dictionaries. Although beyond the scope of this dissertation, one could envision, for instance, augmenting the dictionary to increase its linear span of functions and then pruning it to eliminate the approximation error. Throughout this chapter, we use data-driven methods that are not specifically designed to work with data corrupted with measurement noise. Hence, one might need to pre-process the data before applying the proposed algorithms.

4.2 Parallel Symmetric Subspace Decomposition

Here, we build on the Symmetric Subspace Decomposition (SSD) algorithm introduced in Section 3.3 to address the problem laid out in Section 4.1, i.e., identifying $\tilde{D}$ by means of a parallel strategy, either because the data is not centrally available at a single processor or, even if it is, because its implementation over multiple processors speeds up the identification significantly. Our algorithm, termed Parallel Symmetric Subspace Decomposition[2] (P-SSD), cf. Algorithm 5, has each processor use SSD on its local data and prune its subspace using the estimates of the invariant dictionary communicated by its neighbors. In Algorithm 5, the matrix C_k^i determines the local dictionary of agent i at iteration k in the form of $D(\cdot)C_k^i$. In other words, $\text{span}(D(\cdot)C_k^i)$ is the ith agent's estimate of the maximal Koopman-invariant subspace in $\text{span}(D)$ at iteration k.

The following is an informal description of the algorithm steps:

[*Informal description of P-SSD*:] Given dictionary snapshots $D(X)$, $D(Y)$ distributed over M processors communicating over a network, each agent iteratively: (i) receives its neighbors' estimates of the invariant dictionary, (ii) uses the SSD algorithm to identify the largest invariant subspace (according to the local data) in the intersection of its dictionary and its neighbors' dictionaries, (iii) chooses a basis for the identified subspace as its own dictionary, and (iv) transmits that dictionary to its neighbors.

The proposed P-SSD algorithm builds on the observation that the subspaces identified by SSD are monotone non-increasing with respect to the addition of data (Lemma 3.3.3), i.e., the dimension of the identified subspace does not increase when we add more data.

4.2.1 Equilibria and Termination of P-SSD

Here, we define the concept of equilibria of the P-SSD algorithm and discuss how to detect whether the agents have attained one (we refer to this as termination). Viewing the algorithm as a discrete-time dynamical system, we start by defining the concept of equilibrium.

[2]The function $\text{basis}(\mathcal{A})$ returns a matrix whose columns provide a basis for $\mathcal{A}$. If $\mathcal{A} = \{0\}$, then $\text{basis}(\mathcal{A})$ returns 0. Moreover, $\sharp\text{cols}(0) = 0$.

Algorithm 5 Parallel Symmetric Subspace Decomposition

```
    Inputs: $D(X_i), D(Y_i)$, $i \in \{1, \dots, M\}$
    Output for agent i: $\text{flag}_k^i$, $C_k^i$, $k \in \mathbb{N}_0$    ▷ $D(\cdot)C_k^i$ is the $i$th agent's dictionary at iteration $k$ and $\text{flag}_k^i$ is the flag variable used to detect termination of the algorithm
    Each agent i executes:
 1: $k \leftarrow 0$, $C_0^i \leftarrow I_{N_d}$, $\text{flag}_0^i \leftarrow 0$
 2: while 1 do
 3:     $k \leftarrow k + 1$
 4:     Receive $C_{k-1}^j$, $\forall j \in \mathcal{N}_{\text{in}}^k(i)$
 5:     $D_k^i \leftarrow \text{basis}\Big( \bigcap_{j \in \{i\} \cup \mathcal{N}_{\text{in}}^k(i)} \mathcal{R}(C_{k-1}^j) \Big)$
 6:     $E_k^i \leftarrow \text{SSD}(D(X_i)D_k^i, D(Y_i)D_k^i)$    ▷ Applying Algorithm 1 in Chapter 3
 7:     if $\sharp\text{cols}(D_k^i E_k^i) < \sharp\text{cols}(C_{k-1}^i)$ then
 8:         $C_k^i \leftarrow D_k^i E_k^i$    ▷ The subspace gets smaller
 9:         $\text{flag}_k^i \leftarrow 0$
10:     else
11:         $C_k^i \leftarrow C_{k-1}^i$    ▷ The subspace does not change
12:         $\text{flag}_k^i \leftarrow 1$
13:     end if
14:     Transmit $C_k^i$ to out-neighbors
15: end while
```

Definition 4.2.1. ***(Equilibrium of P-SSD).*** *The agent matrices C_k^i, $i \in \{1, \dots, M\}$ in Algorithm 5 are an equilibrium of P-SSD at time $k \in \mathbb{N}$ if $C_p^i = C_k^i$ for all $p > k$ and all $i \in \{1, \dots, M\}$.* □

Note that, after reaching an equilibrium, the subspaces identified by the agents do not change anymore. The next result shows that the P-SSD algorithm always reaches an equilibrium in a finite number of iterations.

Proposition 4.2.2. ***(P-SSD Reaches an Equilibrium in a Finite Number of Iterations).*** *The P-SSD algorithm executed over any (possibly time-varying) digraph reaches an equilibrium after a finite number of iterations.*

Proof. We prove the result by contradiction. Suppose that the algorithm never reaches an equilibrium, i.e., there exists an increasing sequence $\{k_p\}_{p=1}^{\infty} \subseteq \mathbb{N}_0$ such that, at iteration k_p, at least an agent $j \in \{1, \dots, M\}$ executes Steps 7-9 in Algorithm 5. Hence, $\sharp\text{cols}(C^j_{k_p+1}) < \sharp\text{cols}(C^j_{k_p})$. Consequently, using the fact that the number of columns for the matrix of each agent is non-increasing in time (since they execute either Step 8 or Step 11 of Algorithm 5) and the fact that $k_{p+1} \geq k_p + 1$, one can write

$$\sum_{i=1}^{M} \sharp\text{cols}(C^i_{k_{p+1}}) < \sum_{i=1}^{M} \sharp\text{cols}(C^i_{k_p}), \, \forall p \in \mathbb{N}.$$

Since $\sum_{i=1}^{M} \sharp\text{cols}(C^i_0) = MN_d$, this equation implies that, for $p > MN_d+1$, $\sum_{i=1}^{M} \sharp\text{cols}(C^i_{k_p}) < 0$, a contradiction. □

Since the algorithm runs in parallel, we need a mechanism to detect if the P-SSD algorithm has reached an equilibrium. The flag variables carry out this task.

Proposition 4.2.3. ***(P-SSD Equilibria are Detected by Flag Variables in Time Invariant Digraphs).*** *Given a time-invariant network and under the P-SSD algorithm,* $\text{flag}^i_l = 1$ *for some* $l \in \mathbb{N}$ *and for every* $i \in \{1, \dots, M\}$ *if and only if* $C^i_k = C^i_{l-1}$ *and* $\text{flag}^i_k = 1$ *for every* $i \in \{1, \dots, M\}$ *and every* $k \geq l$.

Proof. The implication from right to left is straightforward (simply put $k = l$). To prove the other implication, we use strong induction. By hypothesis, $\text{flag}^i_l = 1$ for every $i \in \{1, \dots, M\}$ and the P-SSD algorithm has executed Steps 11-12 at iteration l. Hence, $C^i_l = C^i_{l-1}$ for all $i \in \{1, \dots, M\}$. Now suppose that for every $i \in \{1, \dots, M\}$

$$\text{flag}^i_k = 1, \, C^i_k = C^i_{l-1}, \, \forall k \in \{l+1, l+2, \dots, p\}. \tag{4.2}$$

We need to prove that for every $i \in \{1, \ldots, M\}$

$$\text{flag}^i_{p+1} = 1,\ C^i_{p+1} = C^i_{l-1}.$$

Noting that the network is time-invariant, based on Step 5 of Algorithm 5 at iteration $p+1$ and the fact that $C^i_p = C^i_{p-1} = C^i_{l-1}$ for every $i \in \{1, \ldots, M\}$, we have $D^i_{p+1} = D^i_p = D^i_{l-1}$ for every $i \in \{1, \ldots, M\}$. Similarly, $E^i_{p+1} = E^i_p = E^i_{l-1}$ for every $i \in \{1, \ldots, M\}$. As a result,

$$\sharp\text{cols}(D^i_{p+1} E^i_{p+1}) = \sharp\text{cols}(D^i_p E^i_p), \tag{4.3}$$

for all $i \in \{1, \ldots, M\}$. Based on the algorithm and (4.2),

$$\sharp\text{cols}(D^i_p E^i_p) \geq \sharp\text{cols}(C^i_{p-1}) = \sharp\text{cols}(C^i_{l-1}), \tag{4.4}$$

for all $i \in \{1, \ldots, M\}$. Using (4.3), (4.4), and the fact that $\sharp\text{cols}(C^i_p) = \sharp\text{cols}(C^i_{l-1})$, we have

$$\sharp\text{cols}(D^i_{p+1} E^i_{p+1}) \geq \sharp\text{cols}(C^i_p),$$

for all $i \in \{1, \ldots, M\}$. Consequently, the algorithm executes Steps 11-12 for every agent and we have

$$\text{flag}^i_{p+1} = 1,\ C^i_{p+1} = C^i_{l-1},$$

for all $i \in \{1, \ldots, M\}$, which concludes the proof. □

Note that the flags detect an equilibrium with one time-step delay. We say that the P-SSD algorithm has *terminated at step k* if $\text{flag}_k^i = 1$ for all $i \in \{1, \dots, M\}$.

Remark 4.2.4. ***(Termination of the P-SSD Algorithm).*** We point out that one may consider different notions of termination for P-SSD: (i) the algorithm reaches an equilibrium, i.e., the agents continue their calculations but do not change their outputs; (ii) based on Proposition 4.2.3, a user external to the parallel processing hardware terminates the algorithm when $\prod_{i=1}^{M} \text{flag}_k^i = 1$ for some $k \in \mathbb{N}$; (iii) agents use distributed halting algorithms [Lyn97, Pel00] to decide when to terminate based on their output and flags. Here, we only employ (i) and (ii) as appropriate and do not implement (iii) for space reasons. $\square$

4.2.2 Properties of Agents' Matrix Iterates along P-SSD

Here, we characterize important properties of the matrices computed by the agents at each iteration of the P-SSD algorithm. The results provided in this section hold for any (including time-varying) communication network topology. The next result characterizes basic properties of the agents' matrix iterates.

Theorem 4.2.5. ***(Properties of Agents' Matrix Iterates).*** *Under the P-SSD algorithm and for any (possibly time-varying) digraph, for each $i \in \{1, \dots, M\}$,*

(i) for all $k \in \mathbb{N}_0$, C_k^i is zero or has full column rank;

(ii) for all $k \in \mathbb{N}$, $\mathcal{R}(D_k^i E_k^i) = \mathcal{R}(C_k^i)$;

(iii) for all $k \in \mathbb{N}$, $\mathcal{R}(D(X_i) C_k^i) = \mathcal{R}(D(Y_i) C_k^i)$.

Proof. Consider an arbitrary $i \in \{1, \dots, M\}$. We prove (i) by induction. For $k = 0$, $C_0^i = I_{N_d}$. Now, suppose that C_k^i is zero or has full column rank, and let us show the same fact for $k+1$. Note that, if $D_{k+1}^i = 0$ or $E_{k+1}^i = 0$, then $C_{k+1}^i = 0$ and the result follows. Now, suppose that $D_{k+1}^i \neq 0$ and $E_{k+1}^i \neq 0$. Based on definition of basis and Theorem 3.3.1(ii), D_{k+1}^i and E_{k+1}^i have full column rank. Note that the algorithm either executes Step 8 or executes Step 11. Hence, $C_{k+1}^i = D_{k+1}^i E_{k+1}^i$ or $C_{k+1}^i = C_k^i$. Consequently, one can deduce that C_{k+1}^i has full column rank ($C_k^i \neq 0$ since $D_{k+1}^i \neq 0$, hence C_k^i has full column rank), establishing (i).

Next, we show (ii). If the algorithm executes Step 8 at iteration k, then $C_k^i = D_k^i E_k^i$ and consequently $\mathcal{R}(D_k^i E_k^i) = \mathcal{R}(C_k^i)$. Suppose instead that the algorithm executes Step 11, and consequently $C_k^i = C_{k-1}^i$. For the case $C_k^i = 0$, using Step 5, we deduce $D_k^i E_k^i = 0$ and the statement follows. Assume then that $C_k^i \neq 0$ and hence it has full column rank as a result of (i). Consequently, since the condition in Step 7 does not hold and using the definition of $\sharp$cols, we have $D_k^i \neq 0$ and $E_k^i \neq 0$. Also, they have full column rank based on the definition of basis and Theorem 3.3.1(ii). Moreover, $\mathcal{R}(D_k^i E_k^i) \subseteq \mathcal{R}(D_k^i) \subseteq \mathcal{R}(C_{k-1}^i)$ and since the matrices have full column rank, $\sharp\text{cols}(D_k^i E_k^i) \leq \sharp\text{cols}(D_k^i) \leq \sharp\text{cols}(C_{k-1}^i)$. In addition, and based on Step 7, $\sharp\text{cols}(D_k^i E_k^i) \geq \sharp\text{cols}(C_{k-1}^i)$. Hence,

$$\sharp\text{cols}(D_k^i E_k^i) = \sharp\text{cols}(D_k^i) = \sharp\text{cols}(C_{k-1}^i). \tag{4.5}$$

We can use $\sharp\text{rows}(E_k^i) = \sharp\text{cols}(D_k^i)$ and $\sharp\text{cols}(D_k^i E_k^i) = \sharp\text{cols}(E_k^i)$ in conjunction with (4.5) to deduce that E_k^i is square and nonsingular (since it has full column rank). Consequently,

$\mathcal{R}(D_k^i) = \mathcal{R}(D_k^i E_k^i)$. This fact, in combination with $\mathcal{R}(D_k^i) \subseteq \mathcal{R}(C_{k-1}^i)$, yields

$$\mathcal{R}(D_k^i E_k^i) \subseteq \mathcal{R}(C_{k-1}^i). \tag{4.6}$$

Moreover, since D_k^i, E_k^i, and C_{k-1}^i have full column rank, one can use (4.5) and (4.6) to deduce that $\mathcal{R}(D_k^i E_k^i) = \mathcal{R}(C_{k-1}^i)$. Finally, since $C_k^i = C_{k-1}^i$, we have $\mathcal{R}(D_k^i E_k^i) = \mathcal{R}(C_k^i)$, which shows (ii).

To show (iii), from Step 6 of the algorithm and Theorem 3.3.1(ii), for each $i \in \{1, \dots, M\}$ and each $k \in \mathbb{N}$,

$$\mathcal{R}(D(X_i) D_k^i E_k^i) = \mathcal{R}(D(Y_i) D_k^i E_k^i).$$

Based on part (ii) and the fact that $D(X_i), D(Y_i)$ have full column rank, one can deduce $\mathcal{R}(D(X_i) C_k^i) = \mathcal{R}(D(Y_i) C_k^i)$, concluding the proof. $\square$

Remark 4.2.6. ***(Computation of the*** basis ***Function with Finite-Precision Machines).*** Based on Theorem 4.2.5(i), we provide here a practical way of implementing the basis function. Since the matrices C_k^i are either full rank or zero, instead of working with their range space, one can work directly with the matrices themselves. Here, we compute iteratively the output of the basis function for the input matrices. Given the full column-rank matrices A_1, A_2, we find basis$(\mathcal{R}(A_1) \cap \mathcal{R}(A_2))$ using Lemma 3.7.1 by finding the null space of $[A_1, A_2]$ (if one of the matrices is equal to zero, then basis$(\mathcal{R}(A_1) \cap \mathcal{R}(A_2)) = 0$). This procedure can be implemented iteratively for $i \geq 2$ by noting

$$\begin{aligned} &\operatorname{basis}\Big(\bigcap_{j=1}^{i+1} \mathcal{R}(A_i)\Big) \\ &= \operatorname{basis}\bigg(\mathcal{R}\Big(\operatorname{basis}\Big(\bigcap_{j=1}^{i} \mathcal{R}(A_i)\Big)\Big) \bigcap \mathcal{R}(A_{i+1})\bigg). \end{aligned}$$

When implemented on digital computers, this might lead to small errors affecting the null space of $[A_1, A_2]$. To avoid this problem, given the singular values $\sigma_1 \geq \cdots \geq \sigma_l$ of $[A_1, A_2]$ and a tuning parameter $\epsilon_\cap > 0$, we compute k as the minimum integer satisfying $\sum_{j=k}^{l} \sigma_j^2 \leq \epsilon_\cap(\sum_{j=1}^{l} \sigma_j^2)$ and then set $\sigma_k = \cdots = \sigma_l = 0$. The parameter $\epsilon_\cap$ tunes the sensitivity of the implementation. □

Next, we show that the range space identified by an agent at any time step is contained into the range space identified by that agent and its neighbors in the previous time step.

Lemma 4.2.7. ***(Subspace Monotonicity of Agents' Matrix Iterates).*** *Under the P-SSD algorithm and for any (possibly time-varying) digraph, at each iteration $k \in \mathbb{N}$, it holds that $\mathcal{R}(C_k^i) \subseteq \mathcal{R}(C_{k-1}^j)$ for all $i \in \{1, \ldots, M\}$ and all $j \in \{i\} \cup \mathcal{N}_{\text{in}}^k(i)$.*

Proof. Let $i \in \{1, \ldots, M\}$. The case $C_k^i = 0$ is trivial, so consider instead the case $C_k^i \neq 0$. This implies that $D_k^i \neq 0$ and $E_k^i \neq 0$, with both having full column rank based on definition of the basis function and Theorem 3.3.1(ii). Note that $\mathcal{R}(D_k^i E_k^i) \subseteq \mathcal{R}(D_k^i)$ and, by definition, $\mathcal{R}(D_k^i) \subseteq \mathcal{R}(C_{k-1}^j)$ for every $j \in \{i\} \cup \mathcal{N}_{\text{in}}^k(i)$. The result now follows by noting that $\mathcal{R}(D_k^i E_k^i) = \mathcal{R}(C_k^i)$ (cf. Theorem 4.2.5(ii)). □

We conclude this section by showing that the range space of the agents' matrix iterates contain the *SSD subspace*, i.e., the subspace identified by the SSD algorithm (cf. Algorithm 1) when all the data is available at once at a single processor.

Proposition 4.2.8. ***(Inclusion of SSD Subspace by Agents' Matrix Iterates).*** *Let $C_{\text{SSD}} = \text{SSD}\big(D(X), D(Y)\big)$ be the output of the SSD algorithm applied on $D(X), D(Y)$. Under the P-SSD algorithm and for any (possibly time-varying) digraph, at each iteration $k \in \mathbb{N}_0$, it holds that $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_k^i)$ for all $i \in \{1, \dots, M\}$.*

Proof. The case $C_{\text{SSD}} = 0$ is trivial. Suppose that $C_{\text{SSD}} \neq 0$. We prove the argument by induction. Since $C_0^i = I_{N_d}$ for all $i \in \{1, \dots, M\}$, we have

$$\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_0^i) = \mathbb{R}^{N_d}, \, \forall i \in \{1, \dots, M\}.$$

Now, suppose that

$$\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_k^i), \, \forall i \in \{1, \dots, M\}, \tag{4.7}$$

and let us prove the same inclusion for $k+1$. Based on Step 5 in Algorithm 5, one can write

$$\mathcal{R}(D_{k+1}^i) = \bigcap_{j \in \{i\} \cup \mathcal{N}_{\text{in}}^{k+1}(i)} \mathcal{R}(C_k^j), \, \forall i \in \{1, \dots, M\}. \tag{4.8}$$

Using (4.7) and (4.8), we get $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(D_{k+1}^i)$ for every $i \in \{1, \dots, M\}$. Moreover, since both D_{k+1}^i and C_{SSD} have full column rank (the first, based on its definition and the second based on Theorem 3.3.1(ii)), one can find matrices $F_{k+1}^i, i \in \{1, \dots, M\}$ with full column rank such that

$$C_{\mathrm{SSD}} = D^i_{k+1} F^i_{k+1},\ \forall i \in \{1, \ldots, M\}. \tag{4.9}$$

From Theorem 3.3.1(ii), we have $\mathcal{R}(D(X)C_{\mathrm{SSD}}) = \mathcal{R}(D(Y)C_{\mathrm{SSD}})$. Hence, since $\mathrm{rows}\big([D(X_i), D(Y_i)]\big) \subseteq \mathrm{rows}\big([D(X), D(Y)]\big)$ for every $i \in \{1, \ldots, M\}$, we deduce

$$\mathcal{R}(D(X_i)C_{\mathrm{SSD}}) = \mathcal{R}(D(Y_i)C_{\mathrm{SSD}}). \tag{4.10}$$

Combining (4.9) and (4.10), we obtain

$$\mathcal{R}(D(X_i)D^i_{k+1}F^i_{k+1}) = \mathcal{R}(D(Y_i)D^i_{k+1}F^i_{k+1}), \tag{4.11}$$

for every $i \in \{1, \ldots, M\}$. Now, by Step 6 of Algorithm 5, for every $i \in \{1, \ldots, M\}$,

$$E^i_{k+1} = \mathrm{SSD}(D(X_i)D^i_{k+1}, D(Y_i)D^i_{k+1}).$$

Using Theorem 3.3.1(iii) with the dictionary $D(\cdot)D^i_{k+1}$ and data $D(X_i)$, $D(Y_i)$, we deduce

$$\mathcal{R}(F^i_{k+1}) \subseteq \mathcal{R}(E^i_{k+1}), \forall i \in \{1, \ldots, M\}.$$

Combining this with Lemma 3.7.2 and (4.9), we get

$$\mathcal{R}(C_{\mathrm{SSD}}) = \mathcal{R}(D^i_{k+1}F^i_{k+1}) \subseteq \mathcal{R}(D^i_{k+1}E^i_{k+1}), \tag{4.12}$$

for every $i \in \{1, \ldots, M\}$. According to Algorithm 5, either $C^i_{k+1} = C^i_k$ (Step 11) or $C^i_{k+1} = D^i_{k+1} E^i_{k+1}$ (Step 8). Using (4.7) in the former case and (4.12) in the latter, we conclude $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C^i_{k+1})$, for all $i \in \{1, \ldots, M\}$. $\square$

4.3 Equivalence of P-SSD and SSD

In this section, we study under what conditions on the digraph's connectivity, the P-SSD algorithm is equivalent to the SSD algorithm, i.e., it finds the maximal Koopman-invariant subspace contained in the span of the original dictionary. We start by studying the relationship between the matrix iterates and local data of the agents at the beginning and end of a directed path in the digraph.

Proposition 4.3.1. ***(Relationship of Agents' Matrices and Local Data Along Directed Paths).*** *Given an arbitrary constant digraph, let P be a directed path of length l from node j to node i. Then, under the P-SSD algorithm, for each iteration $k \in \mathbb{N}$ and all $q \geq k + l$,*

$$\mathcal{R}(C^i_q) \subseteq \mathcal{R}(C^j_k), \tag{4.13a}$$

$$\mathcal{R}(D(X_j)C^i_q) = \mathcal{R}(D(Y_j)C^i_q). \tag{4.13b}$$

Proof. To prove (4.13a), we label each node on the path P by p_1 through p_{l+1}, with p_1 corresponding to node j and p_{l+1} corresponding to node i (note that a node will have more than one label if it appears more than once in the path, which does not affect the argument).

Based on Lemma 4.2.7, for all $k \in \mathbb{N}$, one can write

$$\mathcal{R}(C_{k+m}^{p_{m+1}}) \subseteq \mathcal{R}(C_{k+m-1}^{p_m}), \, \forall m \in \{1, \ldots, l\}.$$

Using this equation l times yields $\mathcal{R}(C_{k+l}^i) \subseteq \mathcal{R}(C_k^j)$. Moreover, using Lemma 4.2.7 once again for node i, we deduce $\mathcal{R}(C_q^i) \subseteq \mathcal{R}(C_{k+l}^i)$ for every $q \geq k+l$, and the proof follows.

Regarding (4.13b), the case $C_q^i = 0$ is trivial. Suppose then that $C_q^i \neq 0$, with full column rank (cf. Theorem 4.2.5(i)). Using (4.13a) and Theorem 4.2.5(i), we deduce that C_k^j also has full column rank. Therefore, there exists a full column rank matrix F such that

$$C_q^i = C_k^j F. \tag{4.14}$$

From Theorem 4.2.5(iii), we have $\mathcal{R}(D(X_i)C_q^i) = \mathcal{R}(D(Y_i)C_q^i)$. Looking at this equation in a row-wise manner and considering only the signature matrices, one can write

$$\mathcal{R}(D(X_s)C_k^j F) = \mathcal{R}(D(Y_s)C_k^j F), \tag{4.15}$$

where we have used (4.14). From Theorem 4.2.5(iii) for agent j,

$$\mathcal{R}(D(X_j)C_k^j) = \mathcal{R}(D(Y_j)C_k^j).$$

Using this equation and given the fact that $D(X_j)$, $D(Y_j)$, and C_k^j have full column rank, there must exist a square nonsingular matrix K such that

$$D(Y_j)C_k^j = D(X_j)C_k^j K. \tag{4.16}$$

Looking at this equation in a row-wise manner and considering only the signature data matrices,

$$D(Y_s)C_k^j = D(X_s)C_k^j K. \tag{4.17}$$

Using a similar argument for (4.15), one can deduce that there exists a nonsingular square matrix K^* such that

$$D(Y_s)C_k^j F = D(X_s)C_k^j F K^*. \tag{4.18}$$

Multiplying both sides of (4.17) from the right by F and subtracting from (4.18) results in

$$D(X_s)C_k^j(FK^* - KF) = 0.$$

Since $D(X_s)C_k^j$ has full column rank, we deduce $FK^* = KF$. Now, by multiplying both sides of (4.16) from the right by F and replacing KF by FK^*, we can write

$$D(Y_j)C_q^i = D(Y_j)C_k^j F = D(X_j)C_k^j F K^* = D(X_j)C_q^i K^*,$$

where we have used (4.14) twice, which shows the result. □

Next, we show that globally reachable nodes in the digraph determine the SSD subspace in a finite number of iterations.

Theorem 4.3.2. ***(Globally Reachable Nodes Find the SSD Subspace).*** *Given an arbitrary constant digraph, let i be a globally reachable node and define $l = \max_{j\in\{1,\dots,M\}} \operatorname{dist}(j,i)$. Then, under the P-SSD algorithm, $\mathcal{R}(C_k^i) = \mathcal{R}(C_{\text{SSD}})$ for all $k \geq l+1$.*

Proof. If $C_k^i = 0$ for some $k \in \mathbb{N}$, then based on Proposition 4.2.8 and Theorem 3.3.1(ii), we have $C_{\text{SSD}} = 0$ and consequently, $\mathcal{R}(C_k^i) = \mathcal{R}(C_{\text{SSD}})$. Now, suppose that for all $k \in \mathbb{N}$, $C_k^i \neq 0$ and has full column rank (cf. Theorem 4.2.5(i)). Since i is a globally reachable node, for each node $j \in \{1,\dots,M\} \setminus \{i\}$, there exists a directed path with length $l_j = \operatorname{dist}(j,i) < \infty$ to node i. Using Proposition 4.3.1 for $j \in \{1,\dots,M\} \setminus \{i\}$ and Theorem 4.2.5(iii) for agent i , we can write

$$\mathcal{R}(D(X_j)C_{l+1}^i) = \mathcal{R}(D(Y_j)C_{l+1}^i),\ \forall j \in \{1,\dots,M\}.$$

Moreover, since for all $j \in \{1,\dots,M\}$, $D(X_j)$, $D(Y_j)$, and C_{l+1}^i have full column rank, there exist nonsingular square matrices $\{K_j\}_{j=1}^M$ such that

$$D(Y_j)C_{l+1}^i = D(X_j)C_{l+1}^i K_j,\ \forall j \in \{1,\dots,M\}. \tag{4.19}$$

Looking at (4.19) in a row-wise manner and only considering the signature data sets, one can write

$$D(Y_s)C^i_{l+1} = D(X_s)C^i_{l+1}K_j,\ \forall j \in \{1,\dots,M\}.$$

For any $p \neq q \in \{1,\dots,M\}$, we subtract (4.19) evaluated at $j = p$ and at $j = q$ to obtain $D(X_s)C^i_{l+1}(K_q - K_p) = 0$. Since $D(X_s)$ and C^i_{l+1} have full column rank, this implies $K := K_1 = \cdots = K_M$. Looking at (4.19) in a row-wise manner and considering the fact that

$$\bigcup_{j=1}^{M} \operatorname{rows}\big([D(X_j), D(Y_j)]\big) = \operatorname{rows}\big([D(X), D(Y)]\big),$$

we deduce $D(X)C^i_{l+1}K = D(Y)C^i_{l+1}$ and consequently

$$\mathcal{R}(D(X)C^i_{l+1}) = \mathcal{R}(D(Y)C^i_{l+1}).$$

This, together with Theorem 3.3.1(iii), implies $\mathcal{R}(C^i_{l+1}) \subseteq \mathcal{R}(C_{\text{SSD}})$. This inclusion along with Proposition 4.2.8 yields $\mathcal{R}(C^i_{l+1}) = \mathcal{R}(C_{\text{SSD}})$. Finally, for every $k \geq l+1$, Lemma 4.2.7 implies $\mathcal{R}(C^i_k) \subseteq \mathcal{R}(C^i_{l+1}) = \mathcal{R}(C_{\text{SSD}})$, which along with Proposition 4.2.8 yields $\mathcal{R}(C^i_k) = \mathcal{R}(C_{\text{SSD}})$. □

Theorem 4.3.2 shows that the globally reachable nodes identify the SSD subspace. Next, we use this fact to derive guarantees for agreement of the range space of all agents' matrices on the SSD subspace over strongly connected networks (we refer to this as P-SSD reaching consensus).

Theorem 4.3.3. ***(Consensus on SSD Subspace and Time Complexity of P-SSD***

over Strongly Connected Digraphs). *Given a strongly connected digraph with diameter d,*

(i) P-SSD reaches consensus in at most $d+1$ iterations: specifically, for all $k \geq d+1$ and all $i \in \{1, \ldots, M\}$,

$$\mathcal{R}(C_k^i) = \mathcal{R}(C_{\mathrm{SSD}}),$$
$$\mathcal{R}(D(X)C_k^i) = \mathcal{R}(D(Y)C_k^i);$$

(ii) the P-SSD algorithm terminates after at most $d+2$ iterations, i.e, $\mathrm{flag}_{d+2}^i = 1$ *for each $i \in \{1, \ldots, M\}$.*

Proof. Regarding (i), the proof of consensus on the SSD subspace readily follows from Theorem 4.3.2 by noting that any node is globally reachable from any other node through a path with at most d edges. The rest of the statement is a corollary of this fact together with Theorem 3.3.1(ii).

Regarding (ii), one needs to establish that not only the range space of the agents' matrices remains the same but also that the P-SSD algorithm executes Steps 11-12, which means the matrices themselves also remain the same and flags become 1. Note that using (i) we deduce that $\mathcal{R}(C_{d+1}^i) = \mathcal{R}(C_{d+2}^i)$ for all $i \in \{1, \ldots, M\}$. Hence, if $C_{d+1}^i = 0$, then $C_{d+2}^i = 0$, and the P-SSD algorithm does not execute Steps 8-9 but executes Steps 11-12 instead. Consequently, $\mathrm{flag}_{d+2}^i = 1$ for every agent $i \in \{1, \ldots, M\}$. Suppose then that $C_{d+1}^i \neq 0$, with full column rank based on Theorem 4.2.5(i), and consequently $C_{\mathrm{SSD}} \neq 0$ (also with full column rank, cf. Theorem 3.3.1(ii)). Using (i), we then deduce that in Step 5 of the P-SSD algorithm for agent i at iteration $d+2$, we have $\mathcal{R}(D^i_{d+2}) = \mathcal{R}(C_{\text{SSD}})$. Since D^i_{d+2} has full column rank by definition,

$$\mathcal{R}(D(X)D^i_{d+2}) = \mathcal{R}(D(Y)D^i_{d+2}).$$

Looking at this equation in a row-wise manner and considering the fact that $\text{rows}\big([D(X_i), D(Y_i)]\big) \subseteq \text{rows}\big([D(X), D(Y)]\big)$,

$$\mathcal{R}(D(X_i)D^i_{d+2}I) = \mathcal{R}(D(Y_i)D^i_{d+2}I),$$

where I is the identity matrix with appropriate size. This fact, together with definition of E^i_{d+2} in Step 6 of the P-SSD algorithm, implies that $\mathcal{R}(I) \subseteq \mathcal{R}(E^i_{d+2})$ (cf. Theorem 3.3.1(iii)). Consequently, E^i_{d+2} must be a nonsingular square matrix (cf. Theorem 3.3.1(ii)). Hence,

$$\sharp\text{cols}(D^i_{d+2}E^i_{d+2}) = \sharp\text{cols}(D^i_{d+2}) = \sharp\text{cols}(C_{\text{SSD}}) = \sharp\text{cols}(C^i_{d+1}).$$

Consequently, the P-SSD algorithm executes Steps 11-12 and we have $C^i_{d+2} = C^i_{d+1}$ and $\text{flag}^i_{d+2} = 1$ for every agent $i \in \{1, \ldots, M\}$, concluding the proof. $\square$

In general, the P-SSD algorithm might terminate faster than the explicit upper bound given in Theorem 4.3.3. The main reason for this is that each agent depends on performing SSD on its own data and *usually* SSD can identify the maximal Koopman-invariant subspace in the span of dictionary with a moderate amount of data. The next remark shows that the floating point operation (FLOPs) complexity of the P-SSD algorithm is much lower for each processor that the SSD algorithm and hence the P-SSD subspace search runs much faster than SSD.

Remark 4.3.4. ***(Floating Point Operation (FLOP) Complexity of P-SSD).*** For the (standard) case that $N \gg N_d$, let N_i be the number of data snapshots available to agent $i \in \{1, \dots, M\}$. Considering the fact that the most time consuming operation in Algorithm 5 is Step 6, one can use Remark 3.3.2 and deduce that each iteration of the P-SSD algorithm takes $O(N_i N_d^3)$ FLOPs for agent i. Based on Theorem 4.3.3(ii), agent i performs at most $O(N_i N_d^3 d)$ to find the SSD subspace. In case the data is uniformly distributed among the agents, for each $i \in \{1, \dots, M\}$,

$$N_i = O\Big(\sharp\text{rows}(X_s) + \frac{N - \sharp\text{rows}(X_s)}{M}\Big) = O(N/M),$$

where in the last equality we have used $\sharp\text{rows}(X_s) = O(N_d)$ (in fact one can find signature data with $\sharp\text{rows}(X_s) \leq 2N_d$ as a consequence of Assumption 4.1.1). Hence, the FLOPs complexity of each agent with data uniformly distributed across the agents is $O(dNN_d^3/M)$. This gives a factor of d/M reduction when compared to the FLOP complexity $O(NN_d^3)$ of SSD. If the processors are not homogeneous, then the uniform distribution of the data is not optimal. In general, the optimal distribution minimizes the maximum time taken for processors to complete one algorithm iteration, which means that faster processors should get more data. In practice, our simulations show that P-SSD runs drastically faster than the worst-case bound. □

Remark 4.3.5. ***(Communication Complexity of P-SSD).*** Given a strongly connected digraph with M nodes and diameter d, agent i transmits $|\mathcal{N}_{\text{out}}(i)|$ matrices with maximum size of $N_d \times N_d$. Thus, considering the real numbers as basic messages, and using the fact that the algorithm terminates after at most $d+1$ iterations, the communication complexity of the P-SSD algorithm is

$$O(dN_d^2 \sum_{i=1}^{M} |\mathcal{N}_{\text{out}}(i)|) = O(dN_d^2 E),$$

where E is the number of edges in the digraph. In most conventional parallel computing units, the processors communicate through a shared bus. Hence, at iteration k of the P-SSD algorithm, the ith agent only sends one message comprised of the matrix C_k^i to the bus. Therefore, over such networks, the communication complexity reduces to $O(dN_d^2 M)$. □

Remark 4.3.6. ***(Approximated P-SSD).*** The original dictionary might not contain non-trivial Koopman-invariant subspaces. For such cases, one can look for *approximate* informative invariant subspaces by replacing Step 6 of P-SSD with the Approximated-SSD algorithm (cf. Algorithm 4), which approximates Koopman-invariant subspaces given a tuning parameter ϵ. This modification gives rise to the Approximated P-SSD algorithm for which a similar set of results to the ones stated here for the P-SSD algorithm regarding equilibria, finite termination and eventual agreement of the matrix iterates can be established (we omit them for space reasons). The agents' identified subspaces satisfy the accuracy bounds provided in Section 3.5 based on their local data. □

4.4 Robustness Against Packet Drops and Time-Varying Networks

The parallel nature of P-SSD enables the use of parallel and distributed processing hardware, such as GPUs and clusters of processors. The communication between processors required by such hardware is subject to packet drops or communication failures. Moreover, there are instances when some processors are needed for other tasks or go offline temporarily, resulting in time-varying networks. To address these issues, we investigate the robustness of the P-SSD algorithm against packet drops in the communication network. To tackle this problem, we model the network by a time-varying digraph where a dropped packet from agent i to agent j at time $k \in \mathbb{N}$ corresponds to the absence of edge (i, j) from the digraph at that time. The next result shows that if the digraph remains repeatedly jointly strongly connected, the agents executing the P-SSD algorithm reach a consensus on the SSD subspace in finite time.

Theorem 4.4.1. ***(Consensus on SSD Subspace and Finite-Time Convergence of P-SSD over Repeatedly Jointly Strongly Connected Digraphs).*** *Given a time-varying repeatedly jointly strongly connected digraph $\{G_k = (\{1, \dots, M\}, E_k)\}_{k=1}^{\infty}$, the P-SSD algorithm reaches a consensus equilibrium on the SSD subspace in finite time, i.e., there exists $l \in \mathbb{N}$ such that for every iteration $k \geq l$, $C_k^i = C_l^i$ for all $i \in \{1, \dots, M\}$ with*

$$\mathcal{R}(C_l^1) = \mathcal{R}(C_l^2) = \cdots = \mathcal{R}(C_l^M) = \mathcal{R}(C_{\text{SSD}}).$$

Proof. The fact that the P-SSD algorithm reaches an equilibrium is a direct consequence of Proposition 4.2.2, which states that there exists $l \in \mathbb{N}$ such that

$$C_k^i = C_l^i,\ \forall k \geq l,\ \forall i \in \{1, \dots, M\}. \tag{4.20}$$

Hence, we only need to show that the range space of these matrices corresponds to the SSD subspace. Since the digraph is repeatedly jointly strongly connected, there exists a closed "temporal" path after time l that goes through every node of the digraph. By this we mean that there exist L, time instants $l < k_1 < k_2 < \cdots < k_L$ and node labels $p_1, \dots, p_L$ covering all of $\{1, \dots, M\}$ (note that some nodes might correspond to more that one label) such that

$$(p_i, p_{(i \bmod L)+1}) \in E_{k_i},\ \forall i \in \{1, \dots, L\}.$$

Using (4.20), Lemma 4.2.7 at times $\{k_i\}_{i=1}^L$, and the fact that the path is closed, we deduce

$$\mathcal{R}(C_l^{p_1}) \subseteq \mathcal{R}(C_l^{p_L}) \subseteq \cdots \subseteq \mathcal{R}(C_l^{p_2}) \subseteq \mathcal{R}(C_l^{p_1}).$$

Hence, $\mathcal{R}(C_l^{p_1}) = \cdots = \mathcal{R}(C_l^{p_L})$. Since the path goes through every node and (4.20) again, we arrive at the consensus

$$\mathcal{R}(C_k^1) = \cdots = \mathcal{R}(C_k^M),\ \forall k \geq l. \tag{4.21}$$

It remains to show that this consensus is achieved on the SSD subspace. The inclusion

$\mathcal{R}(C_{\mathrm{SSD}}) \subseteq \mathcal{R}(C_k^i)$ for all $i \in \{1,\dots,M\}$ and $k \geq l$ follows from Proposition 4.2.8. To show the other inclusion, first note that if $C_k^i = 0$ for some $i \in \{1,\dots,M\}$ and $k \geq l$, then $C_{\mathrm{SSD}} = 0$ and the proof follows. Suppose then that C_k^i's are nonzero with full column rank (cf. Theorem 4.2.5(i)) for $k \geq l$. Based on Theorem 4.2.5(iii), for every $i \in \{1,\dots,M\}$ and $k \geq l$, we have $\mathcal{R}(D(X_i)C_k^i) = \mathcal{R}(D(Y_i)C_k^i)$. Moreover, using (4.21) and the fact that all matrices C_k^i's have full column rank, we have

$$\mathcal{R}(D(X_i)C_k^j) = \mathcal{R}(D(Y_i)C_k^j),\ \forall i,j \in \{1,\dots,M\},$$

for every $k \geq l$. Using this equality for $j = 1$ and $k = l$,

$$\mathcal{R}(D(X_i)C_l^1) = \mathcal{R}(D(Y_i)C_l^1),\ \forall i \in \{1,\dots,M\}.$$

Hence, for every $i \in \{1,\dots,M\}$ there exists a nonsingular square matrix K_i such that

$$D(X_i)C_l^1 K_i = D(Y_i)C_l^1. \tag{4.22}$$

Now, looking at (4.22) in a row-wise manner, one obtains for the signature dictionary snapshots, $D(X_s)C_l^1 K_i = D(Y_s)C_l^1$, for $i \in \{1,\dots,M\}$, and hence

$$D(X_s)C_l^1(K_i - K_j) = 0,\ \forall i,j \in \{1,\dots,M\}.$$

Since $D(X_s)$ and C_l^1 have full column rank, we have $K := K_1 = \cdots = K_M$. Replacing K_i

by K in (4.22) gives

$$D(X_i)C_l^1 K = D(Y_i)C_l^1, \, \forall i \in \{1, \dots, M\}.$$

Looking at this equation in a row-wise manner and since

$$\bigcup_{i=1}^{M} \text{rows}([D(X_i), D(Y_i)]) = \text{rows}([D(X), D(Y)]),$$

one can write $D(X)C_l^1 K = D(Y)C_l^1$ and consequently

$$\mathcal{R}(D(X)C_l^1) = \mathcal{R}(D(Y)C_l^1).$$

Using now Theorem 3.3.1(iii), we deduce $\mathcal{R}(C_l^1) \subseteq \mathcal{R}(C_{\text{SSD}})$. Using (4.20) and (4.21), we obtain that $\mathcal{R}(C_k^i) = \mathcal{R}(C_k^1) = \mathcal{R}(C_l^1) \subseteq \mathcal{R}(C_{\text{SSD}})$, for all $i \in \{1, \dots, M\}$ and $k \geq l$, and this concludes the proof. $\square$

An interesting interpretation of the result in Theorem 4.4.1 is that, for time-invariant networks subject to failures in the communication links, as long as the agents re-connect at least once within some uniform time period, the P-SSD algorithm identifies the SSD subspace in a finite number of iterations.

Remark 4.4.2. ***(Using the Agents' Matrix Iterates to Find Maximal Koopman-Invariant Subspaces and Eigenfunctions).*** After the agents reach consensus on the SSD subspace, they can use their computed matrix C instead of C_{SSD} to find the dictionary

$\tilde{D}(x) = D(x)C$, for $x \in \mathcal{M}$. Any agent $i \in \{1, \dots, M\}$ can use its P-SSD matrix to find

$$K_{\text{P-SSD}} = \tilde{D}(X_i)^\dagger \tilde{D}(Y_i) = \tilde{D}(X_s)^\dagger \tilde{D}(Y_s).$$

Note that this is also equal to the square matrix K_{SSD} found by SSD, cf. (3.21). Importantly, all the results for SSD subspaces in Chapter 3 are also valid for the new dictionary $\tilde{D}$. $\square$

4.5 Simulation Results

Here, we provide four examples to demonstrate the properties and effectiveness of the P-SSD algorithm[3].

Example 4.5.1. ***(Unstable Nonlinear System).*** Consider the discrete-time system with state $x = [x_1, x_2]$,

$$x_1^+ = 1.2\, x_1 \tag{4.23a}$$

$$x_2^+ = \sqrt[3]{0.8\, x_2^3 + 8\, x_1^2 + 0.1}\,. \tag{4.23b}$$

The system can be transformed into a discrete-time polyflow [JT19] by the nonlinear transformation $[x_1, x_2] \mapsto [x_1, x_2^3]$. We aim to find the informative Koopman eigenfunctions and invariant subspaces. we sample $N = 10^6$ data snapshots with initial conditions in the state space $\mathcal{M} = [-3, 3] \times [-3, 3]$. We use the dictionary D with $N_d = 15$ comprised of all distinct

[3]We intentionally use low-dimensional systems with sparse eigenfunctions to facilitate a complete in-depth presentation of the results. However, we should point out that the results are valid for high-dimensional systems with no conditions on the sparsity of invariant subspaces.

monomials up to degree 4 of the form $\prod_{i=1}^{4} y_i$, with $y_i \in \{1, x_1, x_2\}$ for $i \in \{1, \dots, 4\}$.

To identify the maximal Koopman-invariant subspace in span(D), we implement the SSD algorithm, and the P-SSD algorithm with $M \in \{5, 20, 100\}$ agents communicating according to a directed ring graph with diameter $d = M - 1$. For the P-SSD strategy, we use the first 15 data snapshots in our database as signature data snapshots and distribute the rest of the data evenly among the agents. Both strategies are implemented on a single computer using MATLAB®. We calculate the time elapsed for each iteration of P-SSD as the maximum time taken by agents to execute the algorithm in that iteration using the `tic` and `toc` commands. Since we use finite precision, we apply the approximation provided in Remark 4.2.6 with $\epsilon_{\cap} = 10^{-12}$.

For all $M \in \{5, 20, 100\}$, the P-SSD algorithm reaches the consensus equilibrium after 1 iteration and terminates (based on Remark 4.2.4) after 2 iterations, which is significantly faster than the bounds provided in Theorem 4.3.3. We have also observed in simulations that packet drops do not delay consensus. Both P-SSD and SSD algorithms correctly identify the 7-dimensional subspace spanned by $\{1, x_1, x_1^2, x_1^3, x_2^3, x_1^4, x_1 x_2^3\}$ as the maximal Koopman-invariant subspace in span(D). Table 4.1 shows the time employed by the algorithms to find it. The P-SSD strategy with $M = 5$, $M = 20$, and $M = 100$, is 80%, 96%, and 99% faster than SSD, resp.

Table 4.1: Time elapsed to identify the maximal Koopman-invariant subspace in span(D) associated with the dynamics (4.23).

Method	SSD	P-SSD (5)	P-SSD (20)	P-SSD (100)
Time (ms)	2175	439	88	17

Using the output matrix of any of the agents, cf. Remark 4.4.2, we build the invariant dictionary $\tilde{D}$ and matrix $K_{\text{P-SSD}}$. Table 4.2 shows the Koopman eigenfunctions and their corresponding eigenvalues calculated using the eigendecomposition of $K_{\text{P-SSD}}$. One can verify analytically using (4.23) that the eigenfunctions in Table 4.2 evolve linearly in time. Even though the function x_2 does not belong to the span of the Koopman-invariant subspace, the Koopman eigenfunctions fully capture the behavior of the system since the functions x_1 and x_2^3 do belong. Hence, one can use (2.8) to predict the evolution of any function in the identified subspace or one simply can use

$$\tilde{D}(x^+) = \tilde{D}(x)K_{\text{P-SSD}}, \, \forall x \in \mathcal{M},$$

to describe the behavior of (4.23) in a linear way.

Table 4.2: Identified eigenfunctions and eigenvalues of the Koopman operator associated with the dynamics (4.23).

Eigenfunction	**Eigenvalue**
$\phi_1(x) = 1$	$\lambda_1 = 1$
$\phi_2(x) = x_1$	$\lambda_2 = 1.2$
$\phi_3(x) = x_1^2$	$\lambda_3 = 1.44$
$\phi_4(x) = x_1^3$	$\lambda_4 = 1.728$
$\phi_5(x) = 2\,x_2^3 - 25\,x_1^2 - 1$	$\lambda_5 = 0.8$
$\phi_6(x) = x_1^4$	$\lambda_6 = 2.0736$
$\phi_7(x) = 2\,x_1 x_2^3 - 25\,x_1^3 - x_1$	$\lambda_7 = 0.96$

To demonstrate the effectiveness of our method in long-term predictions, following the Extended Dynamic Mode Decomposition (EDMD) method [WKR15] and Remark 4.4.2, and given an arbitrary dictionary $\mathcal{D}$, we define its linear prediction matrix as $K = \mathcal{D}(X)^\dagger \mathcal{D}(Y)$.

We also define the following relative and angle error functions on a trajectory $\{x(k)\}_{k=0}^{L}$

$$\begin{aligned} E_{\text{relative}}(k) &= \frac{\left\|\mathcal{D}(x(k)) - \mathcal{D}(x_0)K^k\right\|_2}{\|\mathcal{D}(x(k))\|_2} \times 100, \\ E_{\text{angle}}(k) &= \angle\big(\mathcal{D}(x(k)), \mathcal{D}(x_0)K^k\big). \end{aligned} \tag{4.24}$$

We compare the prediction accuracy on the original dictionary D with the dictionary $\tilde{D}$ identified by P-SSD on 1000 trajectories with length $L = 15$. In order to make sure the trajectories remain in the space on which we have trained our methods, we sample the initial conditions from $[-0.1, 0.1] \times [-3, 3]$.

Figure 4.1 shows the median and range between the first and third quartiles of the relative and angle prediction errors for D and $\tilde{D}$. The P-SSD method has zero prediction error since it identifies and predicts the evolutions on Koopman-invariant subspaces. In contrast, the prediction errors on the original dictionary are significantly large, even for short time steps. □

As Example 4.5.1 illustrates, the P-SSD algorithm usually reaches an equilibrium fast and consequently packet drops do not have the opportunity to significantly affect the time to convergence. Also, P-SSD may find the maximal Koopman-invariant subspace even if the agents do not share the signature snapshots. The next example is selected with three goals in mind: (i) showing the importance of signature snapshots, (ii) confirming the tightness of the bound for time complexity in Theorem 4.3.3, and (iii) illustrating the robustness of the P-SSD algorithm against packet drops.

Example 4.5.2. ***(Piecewise Linear System with Packet Drops).*** Given the state

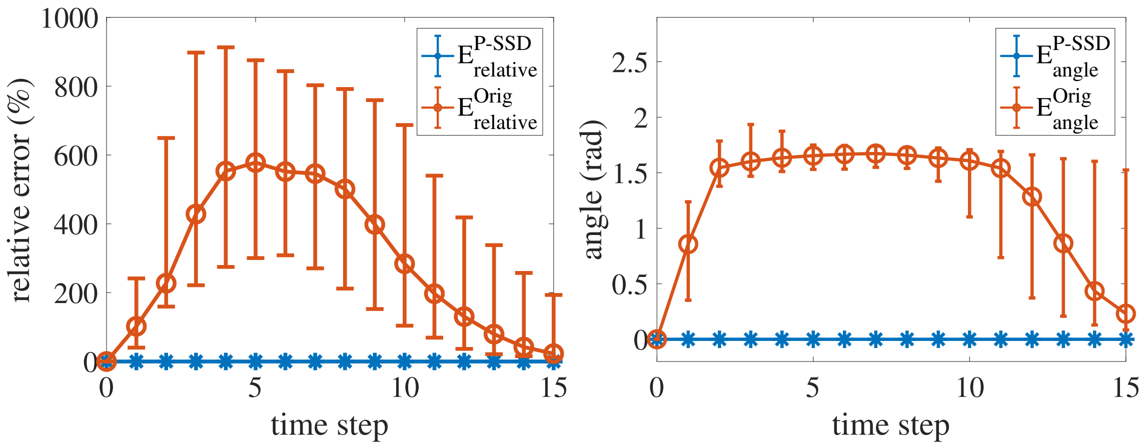

Figure 4.1: Median and range between first and third quartiles of relative (left) and angle (right) prediction errors for the original dictionary and the dictionary identified by P-SSD for system (4.23) on 1000 trajectories of length $L = 15$ with initial conditions randomly selected from $[-0.1, 0.1] \times [-3, 3]$.

space $\mathcal{M} = [-1, 1]^n$, define the sets $\{\mathcal{S}_k\}_{k=1}^n$ by

$$\mathcal{S}_k = \{x \in \mathcal{M} \mid (0 < x_k \leq 1) \wedge (-1 \leq x_j \leq 0, j \neq k)\},$$

where x_i represents the ith element of x. Consider the n-dimensional system $x^+ = T(x)$, where

$$T(x) = \begin{cases} \left(I_n + (k^{-1} - 1)e_k e_k^T\right) x & \text{if } x \in \mathcal{S}_k,\ k \in \{1, \dots, n\}, \\ x & \text{if } x \in \mathcal{M} \setminus \bigcup_{k=1}^n \mathcal{S}_k. \end{cases}$$

Here, e_k is the kth column of the identity matrix I_n. We take $n = 10$ and consider $M = 10$ agents connected according to a directed ring graph with diameter $d = 9$. We use the dictionary D comprised of all $N_d = 66$ monomials of the form $\prod_{i=1}^2 y_i$, where $y_i \in \{x_1, \dots, x_{10}\} \cup \{1\}$ for $i \in \{1, 2\}$.

To perform the simulations, we sample 100 data snapshots with initial conditions in $\mathcal{M} \setminus \bigcup_{k=2}^{n} \mathcal{S}_k$ as our signature snapshots. In addition, we upload additional 1000 data snapshots sampled from $\mathcal{S}_k$ to the kth agent, for each $k \in \{1, \ldots, 10\}$. To check robustness against packet drops, we randomly drop each packet with P percent chance, where we employ $P \in \{0, 10, \ldots, 90\}$. In all cases, the P-SSD algorithm correctly identifies the maximal Koopman-invariant subspace spanned by $\{1, x_1, x_1^2\}$. Figure 4.2 shows the average number of iterations (over 20 simulations) taken by P-SSD to achieve consensus. The plot shows how P-SSD achieves consensus relatively fast even in the presence of 90% packet drops in a directed ring network. The first column of Figure 4.2 indicates that when there is no packet drop, P-SSD reaches consensus in 10 iterations, which is in agreement with the bound provided in Theorem 4.3.3(i).

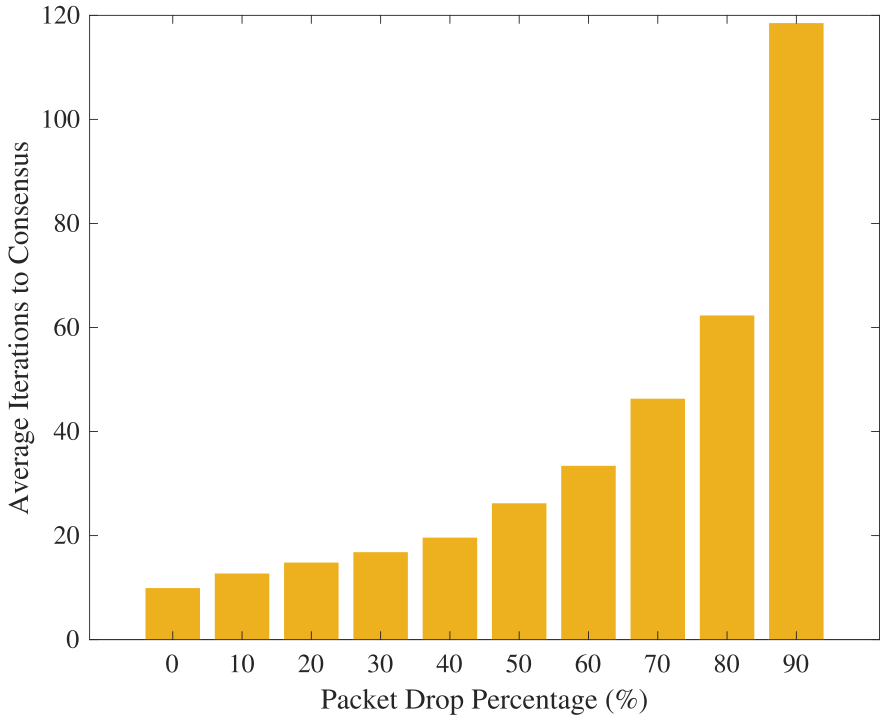

Figure 4.2: Average iterations taken for P-SSD to achieve consensus versus packet drop percentage for Example 4.5.2.

To illustrate the importance of the signature data snapshots, we also perform the simulations without them. In this case, the P-SSD algorithm fails, as it incorrectly identifies the whole space spanned by dictionary D as a Koopman-invariant subspace. This happens because $\text{span}(D)$ is a Koopman-invariant subspace for the system with state space restricted to any of the subsets $\mathcal{S}_k$, $k \in \{1, \dots, 10\}$. Moreover, these different Koopman operators have the same eigenfunctions. However, some of the corresponding eigenvalues are different for different k's. Hence, those eigenfunctions for restricted systems are not eigenfunctions of the Koopman operator for the system defined on the whole state space $\mathcal{M}$. The signature snapshots enable the agents to detect such inconsistencies and prevent the failure of the P-SSD algorithm. □

Example 4.5.3. ***(Van der Pol Oscillator).*** Here, we provide an example of the approximation of Koopman eigenfunctions and invariant subspaces when the original dictionary does not contain exact informative eigenfunctions. Consider the Van der Pol oscillator with state $x = [x_1, x_2]^T$,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + (1 - x_1^2)x_2. \end{aligned} \tag{4.25}$$

To gather data, we run 10^3 simulations for $5s$ with initial conditions uniformly selected from $\mathcal{M} = [-4, 4] \times [-4, 4]$. We sample the trajectory data with time step $\Delta t = 0.05s$ resulting in $N = 10^5$ data snapshots. Moreover, we use the dictionary D with $N_d = 45$ comprised of all distinct monomials up to degree 8 of the form $\prod_{i=1}^{8} y_i$, with $y_i \in \{1, x_1, x_2\}$ for $i \in \{1, \dots, 8\}$. To avoid numerical problems caused by finite-precision errors, we form the matrix $[D(X)^T, D(Y)^T]^T$ and scale each dictionary function such that the norm of columns of the aforementioned matrix become equal. Note that this scaling does not change the functional space spanned by the dictionary since each function in the dictionary is scaled by a nonzero number in $\mathbb{R}$. The communication network is modeled by a complete digraph with $M = 20$ processors resembling a GPU. In addition, we use 1000 snapshots randomly selected from our database as the signature data snapshots and distribute the rest of the data evenly among the processors.

Note that the dictionary D only contains the trivial Koopman eigenfunction $\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$. Consequently, the P-SSD algorithm results in a one-dimensional Koopman-invariant subspace providing exact prediction but no information about the dynamics. To address this issue, we use instead the Approximated P-SSD algorithm presented in Remark 4.3.6 with $\epsilon = 0.005$. Moreover, we set $\epsilon_{\cap} = 0.005$ following Remark 4.2.6. This algorithm reaches a consensus equilibria after 4 iterations identifying a 4-dimensional subspace. It is worth mentioning that we could not implement the Approximated SSD algorithm, cf. Algorithm 4, on the same dataset since it performs SVD on the whole dataset and requires a large memory that is beyond our computational resources (also, performing SVD on such large datasets may result in large round-off errors and inaccurate results).

Using the output matrix of any of the agents, cf. Remark 4.4.2, we find the dictionary $\tilde{D}$ with $\tilde{N}_d = 4$ functions and create the approximated P-SSD matrix as $K_{\text{P-SSD}}^{\text{apprx}} = \tilde{D}(X)^{\dagger}\tilde{D}(Y)$. Moreover, we find the Koopman eigenfunctions approximated by the eigendecomposition of $K_{\text{P-SSD}}^{\text{apprx}}$. Approximated P-SSD finds the only exact Koopman eigenfunction ($\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$) in the span of the original dictionary correctly. Figure 4.3 shows the leading nontrivial approximated eigenfunction corresponding to the largest eigenvalue, which captures the behavior of the Van der Pol oscillator.

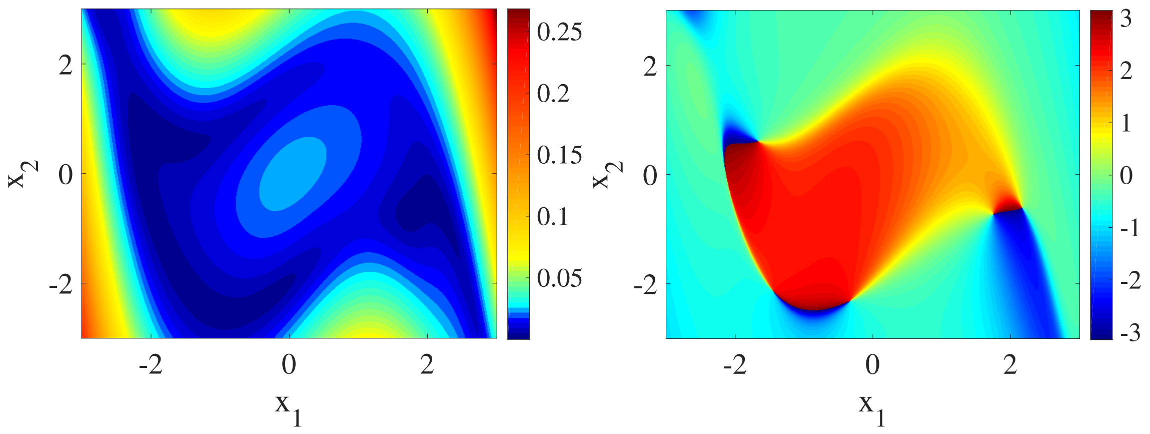

Figure 4.3: Absolute value (left) and angle (right) of the approximated eigenfunction corresponding to eigenvalue $0.9647 + 0.018j$.

To show the advantage of our method in long-term prediction, we use the error functions presented in (4.24) on 1000 trajectories with length $L = 20$ time steps and initial conditions uniformly taken from $[-4, 4] \times [-4, 4]$. Figure 4.4 shows the median and the range

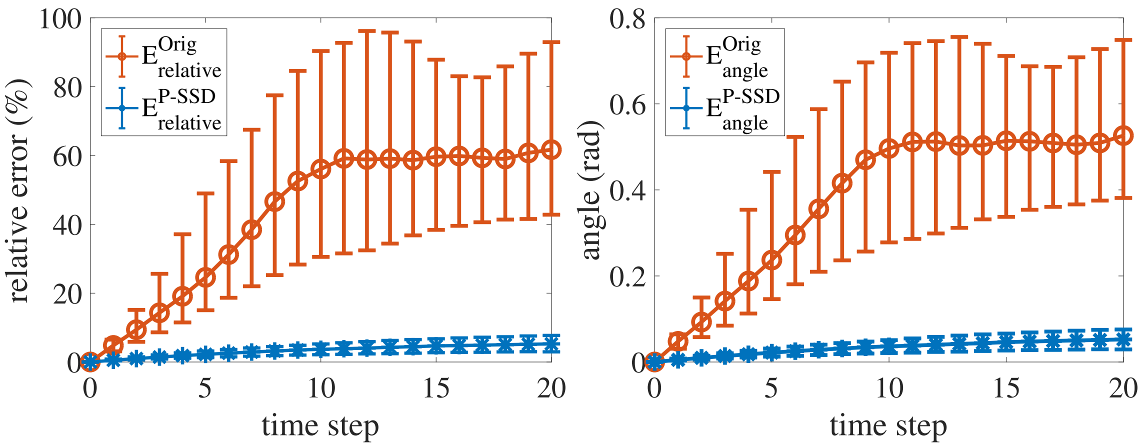

Figure 4.4: Median and range between first and third quartiles of relative (left) and angle (right) errors for the original dictionary and the dictionary identified with Approximated P-SSD for system (4.25) on 1000 trajectories of length $L = 20$ with initial conditions randomly selected from $\mathcal{M}$.

between the first and third quartiles of the predictions errors for the dictionary $\tilde{D}$ identified by Approximated P-SSD and the original dictionary D over the aforementioned trajectories. According to Figure 4.4, Approximated P-SSD has a clear advantage in long-term prediction. After 20 time steps the median of the relative prediction errors for the original dictionary is 60% while the same error for the dictionary identified with Approximated P-SSD is 5%. Moreover, the median of the angle errors after 20 time steps are 0.5 and 0.05 radians for D and $\tilde{D}$, resp. □

Example 4.5.4. ***(Chaotic Lorenz System).*** Consider the chaotic Lorenz system

$$\begin{aligned}\dot{x} &= 10(y-x)\\ \dot{y} &= x(28-z)-y\\ \dot{z} &= xy-(8/3)z,\end{aligned} \tag{4.26}$$

with state $s = [x, y, z]^T$ belonging to state space $\mathcal{M} = [-20, 20] \times [-30, 30] \times [0, 50]$. Our strategy for sampling from $\mathcal{M}$, the number of samples and signature data, and the processor network are similar to Example 4.5.3. We use the dictionary D with $N_d = 84$ comprised of all distinct monomials up to degree 6 of the form $\prod_{i=1}^{6} y_i$, with $y_i \in \{1, x, y, z\}$ for $i \in \{1, \ldots, 6\}$. To avoid numerical problems caused by round-off errors, we scale the dictionary elements as in Example 4.5.3. Note that the dictionary D only contains the trivial Koopman eigenfunction $\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$. Consequently, the P-SSD algorithm results in a one-dimensional Koopman-invariant subspace providing exact prediction but no information about the dynamics. To address this issue, we use instead the Approximated P-SSD algorithm presented in Remark 4.3.6 with $\epsilon = 0.001$. Moreover, we set $\epsilon_{\cap} = 0.001$ following

Remark 4.2.6. This algorithm reaches a consensus equilibria after 3 iterations identifying a 2-dimensional subspace. It is worth mentioning that we could not implement the Approximated SSD algorithm, cf. Algorithm 4 on the same dataset because of its large computational requirements..

Using the output matrix of any of the agents, cf. Remark 4.4.2, we find the dictionary $\tilde{D}$ with $\tilde{N}_d = 2$ and two approximated eigenfunctions in its span. Approximated P-SSD finds the only exact Koopman eigenfunction ($\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$) in the span of the original dictionary correctly. It also approximates another real-valued eigenfunction with eigenvalue $\lambda \approx 0.46$ which predicts that the trajectories converge to an *approximated* invariant set since $|\lambda| < 1$.

To show the advantage of our method in long-term prediction, we use the error functions presented in (4.24) on 1000 trajectories with length $L = 20$ time steps and initial conditions uniformly taken from $\mathcal{M} = [-20, 20] \times [-3, 30] \times [0, 50]$ and compare the prediction accuracy of the dictionary $\tilde{D}$ identified by Approximated P-SSD versus the original dictionary D. Figure 4.5 shows the median and the range between the first and third quartiles of the aforementioned predictions errors, indicating the advantage of P-SSD in long-term prediction. $\square$

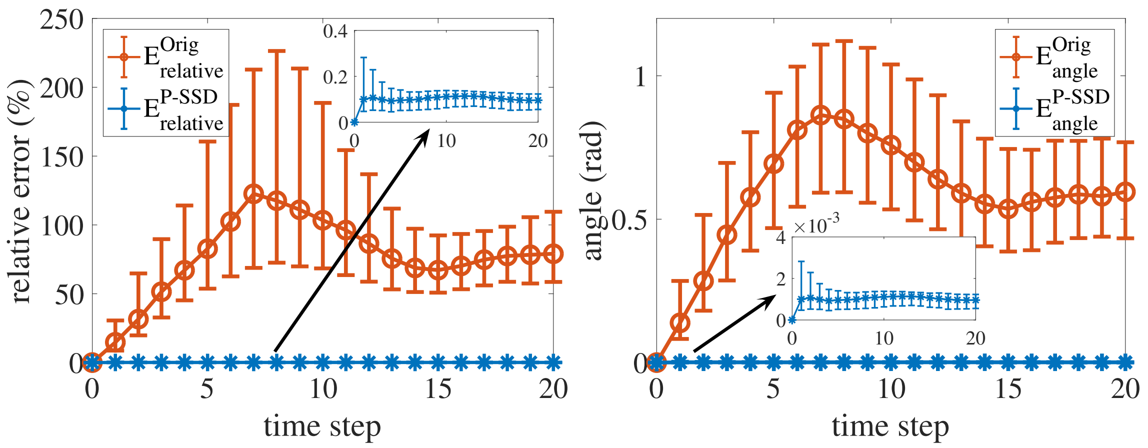

Figure 4.5: Median and range between first and third quartiles of relative (left) and angle (right) errors for dictionary $\tilde{D}$ identified by Approximated P-SSD and the original dictionary D for system (4.26) on 1000 trajectories of length $L = 20$ with initial conditions randomly selected from $\mathcal{M}$.

Acknowledgements

This chapter, in part, is a reprint of the material [HC21b] as it appears in 'Parallel Learning of Koopman Eigenfunctions and Invariant Subspaces For Accurate Long-Term Prediction' by M. Haseli and J. Cortés, in IEEE Transactions on Control of Network Systems 2021. The dissertation author was the primary investigator and author of this paper. The work in this chapter was partially supported by ONR Award N00014-18-1-2828.

Chapter 5

Balancing Accuracy and Expressiveness in Koopman Approximations

In Section 2.4 and Chapter 3, we discussed the Extended Dynamic Mode Decomposition (EDMD) and Symmetric Subspace Decomposition (SSD) methods. These methods can be seen as two extremes of the accuracy-expressiveness trade-off for finite-dimensional Koopman models on subspaces of a particular dictionary's span. Expressiveness corresponds to the ability of the dictionary to describe the evolution of as many observables as possible and accuracy corresponds to the ability to correctly predict their evolution. On the one hand, EDMD provides prediction for all functions in the span of the dictionary leading to maximum expressiveness but there is no guarantee for the accuracy of such predictions. On the other hand, SSD prunes the dictionary's span to find its maximal Koopman-invariant subspace on which the prediction is exact leading to maximum accuracy. However, the pruning by SSD can lead to loss in expressiveness. In this chapter, we aim to explore what is between the two extreme cases of accuracy and expressiveness trade-off.

We first provide a data-driven measure to characterize the prediction accuracy of Koopman-based models on a subspace. Based on the observation that Koopman-invariant subspaces give rise to exact predictions, we reason that prediction accuracy is a function of the degree of invariance of the subspace generated by the dictionary and provide a data-driven measure to characterize invariance proximity. Then we propose an algorithm to iteratively prune the initial functional space to identify a refined dictionary of functions that satisfies the desired level of accuracy while retaining as much of the original expressiveness as possible. We provide a full characterization of the algorithm properties and show that it generalizes both EDMD and SSD methods.

5.1 Problem Statement

Consider [1] the following discrete-time system defined over the state space $\mathcal{M} \subseteq \mathbb{R}^n$

$$x^+ = T(x), \tag{5.1}$$

and let $\mathcal{K}$ be its associated Koopman operator defined on a linear space $\mathcal{F}$ comprised of functions mapping elements in $\mathcal{M}$ to $\mathcal{M}$. Moreover, let $X, Y \in \mathbb{R}^{N\times n}$ be matrices comprised

[1]Some problem elements are similar to the elements used in the previous chapters. Here, we have repeated those notions for the reader's convenience.

of N data snapshots such that

$$y_i = T(x_i),\ i \in \{1, \dots, N\}, \tag{5.2}$$

where x_i^T and y_i^T are ith rows of X and Y. In addition, let $D : \mathcal{M} \to \mathbb{R}^{1\times N_d}$ be a dictionary of N_d functions in $\mathcal{F}$ such that

$$D(\cdot) = [d_1(\cdot), \dots, d_{N_d}(\cdot)]. \tag{5.3}$$

Following Section 2.4, we define the EDMD optimization problem as

$$\underset{K}{\text{minimize}} \|D(Y) - D(X)K\|_F \tag{5.4}$$

which has the closed-form solution

$$K_{\text{EDMD}} = \text{EDMD}(D, X, Y) := D(X)^\dagger D(Y). \tag{5.5}$$

Consequently, given a function $f \in \text{span}(D)$ in the form of $f(\cdot) = D(\cdot)v_f$ for $v_f \in \mathbb{C}^{N_d}$, one can define the predictor function for $\mathcal{K}f$ as

$$\mathfrak{P}_{\mathcal{K}f}(\cdot) = D(\cdot)K_{\text{EDMD}}v_f. \tag{5.6}$$

Moreover, given the eigenvector $v \in \mathbb{C}^{N_d} \setminus \{0\}$ of K_{EDMD} with eigenvalue $\lambda \in \mathbb{C}$, we define the following approximated Koopman eigenfunction with eigenvalue λ as

$$\phi(\cdot) = D(\cdot)v. \tag{5.7}$$

Throughout this chapter, we rely on the following Assumption.

Assumption 5.1.1. ***(Full Rank Dictionary Matrices).*** *$D(X)$ and $D(Y)$ have full column rank.* □

In this chapter, we aim to find dictionaries with elements in span(D) whose span are close to being invariant (with tunable accuracy) under the Koopman operator. We start by noting that one can view the Extended Dynamic Mode Decomposition (EDMD) and Symmetric Subspace Decomposition (SSD) methods described in Sections 2.4 and 3 as the two extreme cases in the trade-off between prediction *accuracy* and dictionary's dimension measuring the *expressiveness* of the identified subspace. Our goal is then to explore the accuracy-expressiveness spectrum in-between the extreme cases of SSD and EDMD. To do this, we seek to provide a formal data-driven characterization of how close a functional space is to being invariant under the Koopman operator (something we refer to as *invariance proximity*). Equipped with this notion, we also aim to develop computational methods that can find finite-dimensional functional spaces that meet a desired level of invariance proximity. We formalize this problem next.

Problem 5.1.2. ***(Characterizing Invariance Proximity of a Subspace and Appropriate Dictionary Identification).*** *Given the original dictionary D defined in* (5.3), *data snapshots matrices* X, Y *gathered from the dynamical system* (5.1) *defined in* (5.2), *and assuming that Assumption 5.1.1 holds, we seek to:*

(i) provide a measure to quantify the invariance proximity of span of any dictionary $\tilde{D}$ *with elements in* $\operatorname{span}(D)$ *solely based on available data* X, Y*;*

(ii) provide an algorithm that finds a dictionary $\tilde{D}$ *with elements in* $\operatorname{span}(D)$ *that meets a desired level of invariance proximity;*

(iii) such that $\operatorname{span}(\tilde{D})$ *contains the maximal Koopman-invariant subspace in* $\operatorname{span}(D)$. $\square$

Requirement (iii) in Problem 5.1.2 ensures the correctness of the algorithmic solution by ensuring the maximal Koopman-invariant subspace and all Koopman eigenfunctions in $\operatorname{span}(D)$ are captured.

5.2 ϵ-Apart Spaces Measure Invariance Proximity

In this section we provide a quantifiable measure for invariance proximity of a subspace by studying the behavior of EDMD with respect to its dictionary. Since the true system dynamics is unknown, this measure must be based on the available data matrices X and Y. To gain a deeper understanding about the behavior of the data-dictionary pair, we offer the following interpretation of the action of the solution K_{EDMD} of the optimization (5.4) as a projection from $\mathcal{R}(D(Y))$ onto $\mathcal{R}(D(X))$. To see this, let $w \in \mathcal{R}(D(Y))$ be a vector of the form of $D(Y)v$. Using (5.5),

$$D(X)K_{\text{EDMD}}v = D(X)D(X)^\dagger D(Y)v = D(X)D(X)^\dagger w = \mathcal{P}_{D(X)}w,$$

where we have used that $D(X)D(X)^\dagger$ is the projection operator on $\mathcal{R}(D(X))$. Using this projection viewpoint alongside (5.4) reveals that the residual error $\|D(Y) - D(X)K_{\text{EDMD}}\|_F$ of EDMD, and consequently its accuracy on the available data, depends of how close the subspaces $\mathcal{R}(D(X))$ and $\mathcal{R}(D(Y))$ are. In fact, note that

- If D spans a Koopman-invariant subspace, we have $\mathcal{R}(D(Y)) = \mathcal{R}(D(X))$ and the residual error of EDMD is equal to zero independently of the data (as long as Assumption 4.1.1 holds). In this case, EDMD captures complete dynamical information about the evolution of the dictionary functions and the predictor (5.6) is exact;

- Instead, if $\mathcal{R}(D(X)) \perp \mathcal{R}(D(Y))$, one can deduce that under Assumption 4.1.1, $K_{\text{EDMD}} = \mathbf{0_{N_d \times N_d}}$ and EDMD captures no information about the dynamics. In particular, the residual error $\|D(Y) - K_{\text{EDMD}}D(X)\|_F = \|D(Y)\|_F$ amounts to 100% prediction error for $D(Y)$.

These observations suggests that the proximity of the vector spaces $\mathcal{R}(D(X))$ and $\mathcal{R}(D(Y))$ can be used as a quantifiable characterization for invariance proximity of span(D) and consequently the prediction accuracy of EDMD. This motivates the following definition.

Definition 5.2.1. ***(ϵ-Apart Subspaces).*** *Given $\epsilon \geq 0$, two vector spaces $S_1, S_2 \subseteq \mathbb{R}^p$ are* ϵ-apart *if $\|\mathcal{P}_{S_1}v - \mathcal{P}_{S_2}v\|_2 \leq \epsilon\|v\|_2$, for all $v \in S_1 \cup S_2$.* □

According to this definition[2], the norm of the error induced by projecting a vector v belonging to one of the subspaces is smaller than $\epsilon\|v\|_2$. Next, we show that this notion fully characterizes equality of spaces with the case $\epsilon = 0$.

[2]Note that, unlike Grassmannians, e.g. [AMS09], there is no restriction on the dimension of the subspaces in Definition 5.2.1.

Lemma 5.2.2. ***(0-apart Subspaces are Equal).*** *Vector spaces $S_1, S_2 \subseteq \mathbb{R}^p$ are 0-apart if and only if $S_1 = S_2$.*

Proof. ($\Rightarrow$): Let $v \in S_1$. By definition, $\|\mathcal{P}_{S_1}v - \mathcal{P}_{S_2}v\|_2 = \|v - \mathcal{P}_{S_2}v\|_2 = 0$, and hence $v = \mathcal{P}_{S_2}v$. Consequently, $v \in S_2$, showing $S_1 \subseteq S_2$. The inclusion $S_2 \subseteq S_1$ can be proved analogously, and we conclude $S_1 = S_2$.

($\Leftarrow$): Since $S_1 = S_2$, for all $v \in S_1 = S_2$, we have $\mathcal{P}_{S_1}v = \mathcal{P}_{S_2}v = v$. Hence, $\|\mathcal{P}_{S_1}v - \mathcal{P}_{S_2}v\|_2 = 0$, for all $v \in S_1 \cup S_2$, and the result follows. □

The next result shows that all subspaces are 1-apart.

Lemma 5.2.3. ***(Any Two Subspaces are 1-apart).*** *Any two vector spaces $S_1, S_2 \subseteq \mathbb{R}^p$ are 1-apart.*

Proof. For any $v \in S_1$, one can write

$$\|\mathcal{P}_{S_1}v - \mathcal{P}_{S_2}v\|_2 = \|v - \mathcal{P}_{S_2}v\|_2 = \|(I - \mathcal{P}_{S_2})v\|_2 \leq \|v\|_2,$$

where in the last equality we have used the fact that $(I - \mathcal{P}_{S_2})$ is the projection operator on the orthogonal complement of S_2. One can write a similar argument for $v \in S_2$, which completes the proof. □

Lemmas 5.2.2-5.2.3 together imply that the range $[0, 1]$ for the parameter ϵ fully characterizes the proximity of any two subspaces. This enables us to use the concept of ϵ-apart subspaces on $D(X)$ and $D(Y)$ as a way to quantify the invariance proximity of span(D) under the Koopman operator associated with the system (5.1). Equipped with this, we reformulate Problem 5.1.2(ii)-(iii) next.

Problem 5.2.4. ***(Balancing Prediction Accuracy and expressiveness).*** *Given the parameter $\epsilon \in [0,1]$, find a dictionary $\tilde{D}$ with elements in* span(D) *such that*

(ii) $\mathcal{R}(\tilde{D}(X))$ *and* $\mathcal{R}(\tilde{D}(Y))$ *are ϵ-apart;*

(iii) span$(\tilde{D})$ *contains the maximal Koopman-invariant subspace in* span(D). □

It is worth mentioning that

$$\epsilon^* = \min\{\epsilon \in [0,1] \mid \mathcal{R}(D(X)), \mathcal{R}(D(Y)) \text{ are } \epsilon\text{-apart}\}$$

captures the invariance proximity, and consequently the prediction accuracy, of D. As a result, if we choose $\epsilon < \epsilon^*$ in Problem 5.2.4, the new dictionary would be smaller than D, leading to a decrease of the expressiveness of the resulting dictionary. Hence, the choice of parameter ϵ strikes a balance between prediction accuracy and expressiveness of the dictionary.

5.3 Tunable Symmetric Subspace Decomposition

In this section, we design and analyze an algorithm, termed Tunable Symmetric Subspace Decomposition (T-SSD), to address Problem 5.2.4.

5.3.1 The T-SSD Algorithm

Given the dictionary D and data snapshots X, Y, the problem of finding a dictionary $\tilde{D}$ such that $\mathcal{R}(\tilde{D}(X))$ and $\mathcal{R}(\tilde{D}(Y))$ are ϵ-apart can be tackled by pruning D. We next describe informally the procedure and then formalize it in Algorithm 6.

[*Informal description*:] The pruning consists of identifying the functions that violate the desired invariance proximity condition and remove them from the dictionary's span. To identify such functions, we define the projection difference matrix (Step 6 in Algorithm 6)

$$G = \mathcal{P}_{D(X)} - \mathcal{P}_{D(Y)} = D(X)D(X)^{\dagger} - D(Y)D(Y)^{\dagger},$$

which is a symmetric matrix with mutually orthogonal eigenvectors spanning $\mathbb{R}^N$ (with corresponding real-valued eigenvalues). Interestingly, if all eigenvalues of G belong to $[-\epsilon, \epsilon]$, then $D(X)$ and $D(Y)$ are ϵ-apart. Otherwise, we focus our attention on the smaller subspace of $\mathbb{R}^N$ defined by

$$\mathcal{W}_\epsilon := \operatorname{span}\{v \in \mathbb{R}^N \mid Gv = \lambda v, \, |\lambda| \leq \epsilon\},$$

corresponding to the span of eigenvectors of G with eigenvalues in $[-\epsilon, \epsilon]$. For practical reasons, we work with a basis for $\mathcal{W}_\epsilon$ (Step 7 in Algorithm 6). Next, we find the largest dictionary $\tilde{D}$ with elements in $\operatorname{span}(D)$ such that $\mathcal{R}(\tilde{D}(X)), \mathcal{R}(\tilde{D}(Y)) \subseteq \mathcal{W}_\epsilon$ (Steps 8-9 in Algorithm 6). There are two possible outcomes:

(i) $\dim \tilde{D} = \dim D$;

(ii) $\dim \tilde{D} < \dim D$.

Scenario (i) indicates that the dictionary D does not need pruning and $\mathcal{R}(D(X)), \mathcal{R}(D(Y))$ are ϵ-apart (Steps 15-17 in Algorithm 6). On the other hand, scenario (ii) leads to a dictionary of lower dimension. However, it is not guaranteed that $\mathcal{R}(\tilde{D}(X))$ and $\mathcal{R}(\tilde{D}(Y))$ are ϵ-apart since $\tilde{D}$ is a different dictionary than D. Consequently, we re-run the process, starting with the definition of G, for the new dictionary $\tilde{D}$. This leads to an iterative implementation that stops when the dictionary cannot be reduced anymore (yielding the desired ϵ-apart subspaces).

The formalization of this procedure yields the Tunable Symmetric Subspace Decomposition (T-SSD)[3] in Algorithm 6. We make the following additional observations regading the use of notation to provide intuition about the algorithm pseudode: (i) we index the internal matrix variables based on the iteration number (this facilitates later the in-depth algebraic analysis); (ii) noting that, at each iteration, the dictionary elements are linear combinations of the elements of the original dictionary, we represent the dictionary at iteration i simply by a matrix C_i, which corresponds to the dictionary $D(\cdot)C_i$; (iii) using the representation in (ii), we do not need to form the dictionary and apply it on the the data matrices X and Y. Instead, the effect of the dictionary at iteration i on the data can be represented as $A_i = D(X)C_i$ and $B_i = D(Y)C_i$.

Algorithm 7 describes the Symmetric-Intersection function in Step 8 of T-SSD: this strategy corresponds to the computation described above of the largest dictionary $\tilde{D}$ such that $\mathcal{R}(\tilde{D}(X))$ and $\mathcal{R}(\tilde{D}(Y))$ belong to the reduced subspace W_ϵ. Similarly to Algorithm 2,

[3]In Algorithms 6-7, the outputs of null(A) and basis(A) are matrices whose columns form orthonormal bases for the null space of A and $\mathcal{R}(A)$, respectively.

Algorithm 6 Tunable Symmetric Subspace Decomposition

```
    Inputs: $D(X), D(Y) \in \mathbb{R}^{N \times N_d}$, $\epsilon \in [0,1]$
 1: Procedure T-SSD($D(X), D(Y), \epsilon$)
 2: Initialization
 3: $i \leftarrow 0$, $A_0 \leftarrow D(X)$, $B_0 \leftarrow D(Y)$, $C_0 \leftarrow I_{N_d}$
 4: while 1 do
 5:     $i \leftarrow i + 1$
 6:     $G_i \leftarrow A_{i-1}A_{i-1}^{\dagger} - B_{i-1}B_{i-1}^{\dagger}$                ▷ projection difference
 7:     $V_i \leftarrow \text{basis}(\text{span}\{v \in \mathbb{R}^N \mid G_i v = \lambda v, \, |\lambda| \leq \epsilon\})$
        ▷ eigenpairs corresponding to small eigenvalues
 8:     $E_i \leftarrow$ Symmetric-Intersection($V_i, A_{i-1}, B_{i-1}$)
        ▷ Find largest dictionary matrices in $V_i$ (Algorithm 7)
 9:     $C_i \leftarrow C_{i-1}E_i$                                   ▷ reduce subspace
10:     $A_i \leftarrow A_{i-1}E_i$, $B_i \leftarrow B_{i-1}E_i$           ▷ calculate new dictionary matrices
11:     if $E_i = 0$ then
12:         return 0                       ▷ ubspace does not exist, returning scalar 0
13:         break
14:     end if
15:     if $\sharp\text{rows}(E_i) \leq \sharp\text{cols}(E_i)$ then
16:         return $C_i$                                  ▷ procedure is complete
17:         break
18:     end if
19: end while
```

instead of actually forming the reduced dictionary, Algorithm 3 uses the matrix-based representation of the dictionary. Next, we explain the steps of the algorithm and the reason behind its naming. Given input matrices V, A, and B, Step 6 in Algorithm 7 identifies W_A such that $\mathcal{R}(AW_A) = \mathcal{R}(V) \cap \mathcal{R}(A)$ (see Lemma 3.7.1). Then, again in Step 13 the algorithm (by virtue of Lemma 3.7.1) finds the matrix Z_B such that $\mathcal{R}(BW_AZ_B) = \mathcal{R}(V) \cap \mathcal{R}(BW_A)$. The output matrix $E := \text{basis}(W_AZ_B)$ (cf. Step 13) then specifies the largest subspaces $\mathcal{R}(AE)$, $\mathcal{R}(BE)$ both belonging to $\mathcal{R}(V)$. Note the symmetry in this specification: if a linear combination of the columns of A is in $\mathcal{R}(V)$, then the same linear combination of columns of B belongs to $\mathcal{R}(V)$. Moreover, Algorithm 7 breaks and returns 0 if any of the aforementioned intersections only contain the zero vector (Steps 2-4 and Steps 7-9).

Algorithm 7 Symmetric Intersection

```
    Inputs: $V \in \mathbb{R}^{n\times m}$ and $A, B \in \mathbb{R}^{n\times p}$
 1: Procedure Symmetric-Intersection($V, A, B$)
 2: if null($[V, A]$) $= \emptyset$ then
 3:     return 0
 4:     break
 5: else
 6:     $\begin{bmatrix} W_V \\ W_A \end{bmatrix} \leftarrow$ null($[V, A]$)          $\triangleright$ $\sharp$cols($V$) $=$ $\sharp$rows($W_V$), $\sharp$cols($A$) $=$ $\sharp$rows($W_A$)
 7:     if null($[V, BW_A]$) $= \emptyset$ then
 8:         return 0
 9:         break
10:     end if
11:     $\begin{bmatrix} Z_V \\ Z_B \end{bmatrix} \leftarrow$ null($[V, BW_A]$)          $\triangleright$ $\sharp$cols($V$) $=$ $\sharp$rows($Z_V$), $\sharp$cols($BW_A$) $=$ $\sharp$rows($Z_B$)
12: end if
13: return basis($W_A Z_B$)          $\triangleright$ returning an orthogonal basis
```

Remark 5.3.1. ***(Implementation of Algorithm 7 on Finite-Precision Computers).*** The accuracy of the implementation of Algorithm 7 depends on the calculation of the null space of several matrices, which might be sensitive to round-off errors. To circumvent this issue, one can set sufficiently small (according to a desired accuracy level) singular values of the matrices to zero. □

5.3.2 Basic Properties of T-SSD

Our end goal now is to show that the T-SSD algorithm solves Problem 5.2.4 and unveil its relationship with the EDMD and SSD methods. In order to do so, we establish here several basic algorithm properties.

Proposition 5.3.2. ***(Properties of*** Symmetric-Intersection***).*** *Let matrices V, A, B have* *full column rank and* $E = \text{Symmetric-Intersection}(V, A, B)$*. Then,*

(i) $E = 0$ *or* $E^T E = I$*;*

(ii) $\mathcal{R}(AE), \mathcal{R}(BE) \subseteq \mathcal{R}(V)$*;*

(iii) E *is maximal, i.e., any nonzero matrix* F *such that* $\mathcal{R}(AF), \mathcal{R}(BF) \subseteq \mathcal{R}(V)$ *satisfies* $\mathcal{R}(F) \subseteq \mathcal{R}(E)$*.*

Proof. (i) There are three ways for Algorithm 7 to terminate. If the algorithm executes Steps 2-4 or Steps 7-9, we have $E = 0$ by definition. Otherwise, the algorithm executes Step 13. Hence, noting that W_A and Z_B exist and the basis function returns an orthonormal basis for $W_A Z_B$, one can conclude $E^T E = I$.

(ii) The case $E = 0$ is trivial. Suppose that $E \neq 0$ and hence has full column rank according to part (i). By definition, $\mathcal{R}(E) = \mathcal{R}(W_A Z_B)$. Consequently, based on Step 11 of the algorithm and using Lemma 3.7.1, we deduce

$$\mathcal{R}(BE) = \mathcal{R}(BW_A Z_B) = \mathcal{R}(BW_A) \cap \mathcal{R}(V) \subseteq \mathcal{R}(V), \tag{5.8}$$

where in the first equality, we used Lemma 3.7.2. Moreover, from the definition of E, one can deduce that $\mathcal{R}(E) \subseteq \mathcal{R}(W_A)$. In addition, based on Lemma 3.7.2, we have $\mathcal{R}(AE) \subseteq \mathcal{R}(AW_A)$. Using the previous inequality in conjunction with Lemma 3.7.1 applied to Step 6 of the algorithm, one can write

$$\mathcal{R}(AE) \subseteq \mathcal{R}(AW_A) = \mathcal{R}(A) \cap \mathcal{R}(V) \subseteq \mathcal{R}(V), \tag{5.9}$$

which in conjunction with (5.8) concludes the proof of (ii).

(iii) Without loss of generality, we assume that F has full column rank (if that is not the case, one can consider another matrix $\bar{F}$ with full column rank such that $\mathcal{R}(F) = \mathcal{R}(\bar{F})$). Since $\mathcal{R}(AF) \subseteq \mathcal{R}(V)$, we have $\mathcal{R}(AF) \subseteq \mathcal{R}(A) \cap \mathcal{R}(V)$, which leads to $\mathcal{R}(AF) \subseteq \mathcal{R}(AW_A)$ based on (5.9). Moreover, one can use Lemma 3.7.2 to deduce that $\mathcal{R}(F) \subseteq \mathcal{R}(W_A)$. Since F and W_A both have full column rank, there exists F_W with full column rank such that

$$F = W_A F_W. \tag{5.10}$$

Considering that $\mathcal{R}(BF) \subseteq \mathcal{R}(V)$ and $\mathcal{R}(BW_AF_W) \subseteq \mathcal{R}(BW_A)$ in combination with (5.10), we deduce $\mathcal{R}(BF) = \mathcal{R}(BW_AF_W) \subseteq \mathcal{R}(BW_A) \cap \mathcal{R}(V) = \mathcal{R}(BW_AZ_B)$, where the last equality follows from (5.8). Based on Lemma 3.7.2, we deduce $\mathcal{R}(F) \subseteq \mathcal{R}(W_AZ_B) = \mathcal{R}(E)$. $\square$

Next, we show that T-SSD terminates after a finite number of iterations.

Proposition 5.3.3. ***(Finite-time Termination of T-SSD Algorithm).*** *The T-SSD algorithm terminates after at most N_d iterations.*

Proof. We reason by contradiction. Suppose that the algorithm does not terminate before iteration $N_d + 1$. Hence, the algorithm does not execute Steps 12-13 or Steps 16-17 in the first N_d iterations. Therefore, the conditions in Steps 11 and 15 do not hold. Consequently, using Proposition 5.3.2(i), one can write

$$\sharp\text{rows}(E_i) > \sharp\text{rows}(E_i) - 1 \geq \sharp\text{cols}(E_i), \tag{5.11}$$

for all $i \in \{1, \ldots, N_d\}$. In addition, based on the definition of the E_i's, one can deduce $\sharp\text{cols}(E_i) = \sharp\text{rows}(E_{i+1})$, for all $i \in \{1, \ldots, N_d\}$. Combining this with (5.11) leads to $\sharp\text{rows}(E_1) \geq \sharp\text{cols}(E_{N_d}) + N_d$. This fact together with $\sharp\text{rows}(E_1) = N_d$ and $\sharp\text{cols}(E_{N_d}) = \sharp\text{cols}(C_{N_d})$ (cf. Step 9) implies that $\sharp\text{cols}(C_{N_d}) \leq 0$, contradicting $\sharp\text{cols}(C_{N_d}) \geq 1$. $\square$

Next, we study basic properties of the internal matrices of the T-SSD algorithm.

Lemma 5.3.4. ***(Properties of T-SSD Matrices).*** *Let the T-SSD algorithm terminate in L time steps. Then,*

(i) $\forall i \in \{0, \ldots, L-1\}$, $\mathcal{R}(C_{i+1}) \subseteq \mathcal{R}(C_i)$;

(ii) $\forall i \in \{0, \ldots, L-1\}$, $C_i^T C_i = I$;

(iii) $C_L = \mathbf{0}$ *or* $C_L^T C_L = I$,

where C_i denotes T-SSD's ith internal matrix, cf. Algorithm 6.

Proof. (i) According to Step 9 of the algorithm, $C_{i+1} = C_i E_{i+1}$. Hence, $\mathcal{R}(C_{i+1}) = \mathcal{R}(C_i E_{i+1}) \subseteq \mathcal{R}(C_i)$.

(ii) For $i = 0$, the result holds by definition. Moreover, since the algorithm does not terminate until iteration L, it does not execute Steps 12-13 in iterations $\{1, \ldots, L-1\}$. Hence, $E_i \neq 0$ and based on Proposition 5.3.2(i), we have

$$E_i^T E_i = I, \, \forall i \in \{1, \ldots, L-1\}. \tag{5.12}$$

Moreover, from Step 9, $C_i = C_0 E_1 E_2 \cdots E_i$, $\forall i \in \{1, \ldots, L-1\}$. This in conjunction with (5.12) and $C_0 = I_{N_d}$, implies $C_i^T C_i = I$ for all $i \in \{1, \ldots, L-1\}$, as claimed.

(iii) Note that $C_L = C_{L-1}E_L$. Based on Proposition 5.3.2(i), either $E_L = 0$ or $E_L^T E_L = I$. In the former case, we have $C_L = \mathbf{0}$. In the latter case, $C_L^T C_L = C_{L-1}^T E_L^T E_L C_{L-1} = C_{L-1}^T C_{L-1} = I$, where in the last equality we used (ii). $\square$

For convenience, let

$$C_{\text{T-SSD}} := \text{T-SSD}(D(X), D(Y), \epsilon), \tag{5.13}$$

denote the output of the T-SSD algorithm. This leads to the definition of the T-SSD dictionary

$$D_{\text{T-SSD}}(\cdot) := D(\cdot)C_{\text{T-SSD}}. \tag{5.14}$$

To extract the dynamical information associated with the Koopman operator on $\text{span}(D_{\text{T-SSD}})$, we use EDMD. According to (5.5), we find the T-SSD prediction matrix as

$$K_{\text{T-SSD}} := \text{EDMD}(D_{\text{T-SSD}}, X, Y) = D_{\text{T-SSD}}(X)^\dagger D_{\text{T-SSD}}(Y). \tag{5.15}$$

We can also define approximated Koopman eigenfunctions according to (5.7) using the eigendecomposition of $K_{\text{T-SSD}}$ and the dictionary $D_{\text{T-SSD}}$. In addition, following (5.6), given any function $f \in \text{span}(D_{\text{T-SSD}})$ described by $f(\cdot) = D_{\text{T-SSD}}(\cdot)w$, we can define the T-SSD predictor of $\mathcal{K}f$ on $\operatorname{span}(D_{\text{T-SSD}})$ as

$$\mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(\cdot) = D_{\text{T-SSD}}(\cdot)K_{\text{T-SSD}}w. \tag{5.16}$$

Remark 5.3.5. ***(Computational Complexity of T-SSD).*** Given N data snapshots and N_d dictionary functions, and considering the complexity of scalar operations as $O(1)$, the most time-consuming step of Algorithm 6 is Step 7, which requires the eigendecomposition of an $N \times N$ matrix and takes $O(N^3)$ operations. Based on Proposition 5.3.3, the algorithm terminates after at most N_d iterations, resulting in a total complexity of $O(N^3 N_d)$. □

5.4 T-SSD Balances Accuracy and Expressiveness

In this section we show that the output of T-SSD balances prediction accuracy and expressiveness as prescribed by the design parameter $\epsilon \in [0, 1]$.

5.4.1 T-SSD Identifies ϵ-Apart Subspaces

Here, we show that T-SSD solves Problem 5.2.4(ii).

Theorem 5.4.1. ***(T-SSD Output Subspaces are ϵ-Apart).*** *$\mathcal{R}(D_{\text{T-SSD}}(X))$ and $\mathcal{R}(D_{\text{T-SSD}}(Y))$ are ϵ-apart.*

Proof. Let $L \leq N_d$ be the number of iterations for convergence of the T-SSD algorithm (cf. Proposition 5.3.3). Based on Proposition 5.3.2(i), we have $E_L = 0$ or $E_L^T E_L = I$. With the notation of Algorithm 6, in the former case, the algorithm executes Steps 12-13 at iteration

L and consequently $C_{\text{T-SSD}} = 0$. Therefore,

$$\mathcal{R}(D_{\text{T-SSD}}(X)) = \mathcal{R}(D_{\text{T-SSD}}(Y)) = \{\mathbf{0}_{N\times 1}\},$$

and the result holds trivially. Now, suppose that $E_L^T E_L = I$. Hence, E_L has full column rank and consequently $\sharp\text{rows}(E_L) \geq \sharp\text{cols}(E_L)$. However, since the algorithm executes Steps 16-17, the condition in Step 15 holds and one can write $\sharp\text{rows}(E_L) = \sharp\text{cols}(E_L)$. Therefore, since E_L has full column rank, it is a nonsingular square matrix and

$$\mathcal{R}(C_L) = \mathcal{R}(C_{L-1}E_L) = \mathcal{R}(C_{L-1}), \tag{5.17a}$$

$$\mathcal{R}(A_L) = \mathcal{R}(A_{L-1}E_L) = \mathcal{R}(A_{L-1}), \tag{5.17b}$$

$$\mathcal{R}(B_L) = \mathcal{R}(B_{L-1}E_L) = \mathcal{R}(B_{L-1}). \tag{5.17c}$$

At iteration L, one can use Steps 6 and 7 in conjunction with the fact that the eigenvectors of G_L are mutually orthogonal to write

$$\begin{aligned} \|G_L v\|_2 &= \|A_{L-1}A_{L-1}^{\dagger}v - B_{L-1}B_{L-1}^{\dagger}v\|_2 \\ &= \|\mathcal{P}_{A_{L-1}}v - \mathcal{P}_{B_{L-1}}v\|_2 \leq \epsilon\|v\|_2, \end{aligned} \tag{5.18}$$

for all $v \in \mathcal{R}(V_L)$. Moreover, based on definition of E_L and Proposition 5.3.2(ii),

$$\mathcal{R}(A_{L-1}E_L), \mathcal{R}(B_{L-1}E_L) \subseteq \mathcal{R}(V_L). \tag{5.19}$$

Consequently, using $A_L = D(X)C_L$ and $B_L = D(Y)C_L$, and equations (5.17)-(5.19), we deduce

$$\begin{aligned}\|\mathcal{P}_{D(X)C_L} v - \mathcal{P}_{D(Y)C_L} v\|_2 &= \|\mathcal{P}_{A_L} v - \mathcal{P}_{B_L} v\|_2 \\ &= \|\mathcal{P}_{A_{L-1}} v - \mathcal{P}_{B_{L-1}} v\|_2 \leq \epsilon \|v\|_2,\end{aligned}$$

for all $v \in \mathcal{R}(D(X)C_L) \cup \mathcal{R}(D(Y)C_L)$. Since $C_L = C_{\text{T-SSD}}$, and given the definition (5.14) of the T-SSD dictionary, this can be rewritten as $\|\mathcal{P}_{D_{\text{T-SSD}}(X)} v - \mathcal{P}_{D_{\text{T-SSD}}(Y)} v\|_2 \leq \epsilon \|v\|_2$, for all $v \in \mathcal{R}(D_{\text{T-SSD}}(X)) \cup \mathcal{R}(D_{\text{T-SSD}}(Y))$. Hence, $\mathcal{R}(D_{\text{T-SSD}}(X))$ and $\mathcal{R}(D_{\text{T-SSD}}(Y))$ are ϵ-apart. □

We next build on Theorem 5.4.1 to characterize the accuracy of predictions (5.16) for any function in $\text{span}(D_{\text{T-SSD}})$ on the available data.

Theorem 5.4.2. ***(Relative Root Mean Square Error (RRMSE) of Koopman Predictions by T-SSD are Bounded by ϵ).*** *For any function $f \in \text{span}(D_{\text{T-SSD}})$,*

$$\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} |\mathcal{K}f(x_i) - \mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(x_i)|^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} |\mathcal{K}f(x_i)|^2}} \leq \epsilon \tag{5.20}$$

where x_i^T is the ith row of X and $\mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}$ is defined in (5.16).

Proof. For convenience, we use the compact notation $\tilde{D}$ to refer to $D_{\text{T-SSD}}$ throughout the proof. We first prove the statement for real-valued functions in $\text{span}(\tilde{D})$. Let $f(\cdot) = \tilde{D}(\cdot)w$ with $w \in \mathbb{R}^{\sharp\text{cols}(C_{\text{T-SSD}})}$. From Theorem 5.4.1, one can write $\|(\mathcal{P}_{\tilde{D}(Y)} - \mathcal{P}_{\tilde{D}(X)})v\|_2 \leq \epsilon \|v\|_2$, for all $v \in \mathcal{R}(\tilde{D}(X)) \cup \mathcal{R}(\tilde{D}(Y))$. One can rewrite this equation as

$$\|(\tilde{D}(Y)\tilde{D}(Y)^\dagger - \tilde{D}(X)\tilde{D}(X)^\dagger)v\|_2 \le \epsilon \|v\|_2, \tag{5.21}$$

for all $v \in \mathcal{R}(\tilde{D}(X)) \cup \mathcal{R}(\tilde{D}(Y))$. In addition, using equations (5.15) and (5.16), and the fact that $\mathcal{K}f(x_i) = f \circ T(x_i) = f(y_i)$ for all $i \in \{1, \ldots, N\}$, one can write

$$\begin{aligned}\sqrt{\sum_{i=1}^{N} \left|\mathcal{K}f(x_i) - \mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(x_i)\right|^2} &= \|(\tilde{D}(Y) - \tilde{D}(X)K_{\text{T-SSD}})w\|_2 \\ &= \|(\tilde{D}(Y) - \tilde{D}(X)\tilde{D}(X)^\dagger\tilde{D}(Y))w\|_2 \\ &= \|\big(\tilde{D}(Y)\tilde{D}(Y)^\dagger - \tilde{D}(X)\tilde{D}(X)^\dagger\big)\tilde{D}(Y)w\|_2,\end{aligned}$$

where we have used $\tilde{D}(Y) = \tilde{D}(Y)\tilde{D}(Y)^\dagger\tilde{D}(Y)$ in the last equality. Moreover, since $\tilde{D}(Y)w \in \mathcal{R}(\tilde{D}(X)) \cup \mathcal{R}(\tilde{D}(Y))$, one can use this equation in conjunction with (5.21) to write

$$\begin{aligned}\sqrt{\sum_{i=1}^{N} \left|\mathcal{K}f(x_i) - \mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(x_i)\right|^2} &\le \epsilon \|\tilde{D}(Y)w\|_2 \\ &= \epsilon \sqrt{\sum_{i=1}^{N} |\mathcal{K}f(x_i)|^2}.\end{aligned}$$

Scaling both sides by $N^{-\frac{1}{2}}$ yields (5.20) for real-valued functions in $\text{span}(\tilde{D})$.

For the complex-valued case, let $f(\cdot) = \tilde{D}(\cdot)w$ with $w = w_{\text{Re}} + jw_{\text{Im}}$, $w_{\text{Re}}, w_{\text{Im}} \in \mathbb{R}^{\sharp\text{cols}(C_{\text{T-SSD}})}$ and $w_{\text{Im}} \neq 0$.

Consider the decompositions of f and $\mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}$ as $f(\cdot) = f_{\text{Re}}(\cdot) + jf_{\text{Im}}(\cdot)$ and

$\mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(\cdot) = \mathfrak{P}_{\mathcal{K}f_{\text{Re}}}^{\text{T-SSD}}(\cdot) + j\mathfrak{P}_{\mathcal{K}f_{\text{Im}}}^{\text{T-SSD}}(\cdot)$, where

$$\begin{aligned} f_{\text{Re}}(\cdot) &= \tilde{D}(\cdot)w_{\text{Re}}, & f_{\text{Im}}(\cdot) &= \tilde{D}(\cdot)w_{\text{Im}}, \\ \mathfrak{P}_{\mathcal{K}f_{\text{Re}}}^{\text{T-SSD}}(\cdot) &= \tilde{D}(\cdot)K_{\text{T-SSD}}w_{\text{Re}}, & \mathfrak{P}_{\mathcal{K}f_{\text{Im}}}^{\text{T-SSD}}(\cdot) &= \tilde{D}(\cdot)K_{\text{T-SSD}}w_{\text{Im}}. \end{aligned} \tag{5.22}$$

Using (5.20) for the real-valued functions in (5.22),

$$\begin{aligned} \sum_{i=1}^{N} |\mathcal{K}f_{\text{Re}}(x_i) - \mathfrak{P}_{\mathcal{K}f_{\text{Re}}}^{\text{T-SSD}}(x_i)|^2 \leq \epsilon^2 \sum_{i=1}^{N} |\mathcal{K}f_{\text{Re}}(x_i)|^2, \\ \sum_{i=1}^{N} |\mathcal{K}f_{\text{Im}}(x_i) - \mathfrak{P}_{\mathcal{K}f_{\text{Im}}}^{\text{T-SSD}}(x_i)|^2 \leq \epsilon^2 \sum_{i=1}^{N} |\mathcal{K}f_{\text{Im}}(x_i)|^2. \end{aligned}$$

By adding these two inequalities, using (5.22), and noting that $|g|^2 = |g_{\text{Re}}|^2 + |g_{\text{Im}}|^2$ for $g = g_{\text{Re}} + jg_{\text{Im}}$, one can write

$$\sum_{i=1}^{N} |\mathcal{K}f(x_i) - \mathfrak{P}_{\mathcal{K}f}^{\text{T-SSD}}(x_i)|^2 \leq \epsilon^2 \sum_{i=1}^{N} |\mathcal{K}f(x_i)|^2,$$

and (5.20) follows. □

Theorem 5.4.2 ensures that each member of the vector space of functions identified by T-SSD has prediction error bounded by the accuracy parameter ϵ.

Remark 5.4.3. ***(T-SSD Bounds the Relative L_2-norm Error of Koopman Predictions under Empirical Measure by ϵ).*** Given the functions in span(D) and their composition with T are measurable, consider the empirical measure $\mu = \frac{1}{N}\sum_{k=1}^{N} \delta_{x_k}$ where

δ_{x_k} is the Dirac measure at the kth row of X. Then Theorem 5.4.2 can be interpreted as

$$\frac{\|\mathcal{K}f - \mathfrak{P}^{\text{T-SSD}}_{\mathcal{K}f}\|_{L_2}}{\|\mathcal{K}f\|_{L_2}} \leq \epsilon, \ \forall f \in \operatorname{span}(D_{\text{T-SSD}}),$$

where the L_2-norm is calculated based on the empirical measure μ. $\square$

5.4.2 T-SSD Captures Maximal Koopman-Invariant Subspace

Here, we show that T-SSD also solves Problem 5.2.4(iii). To do this, we study the relationship of the algorithm with Koopman eigenfunctions and invariant subspaces. We first show that the T-SSD matrices capture the maximal Koopman-invariant subspaces in the span of the original dictionary D.

Theorem 5.4.4. ***(T-SSD Matrices Capture the Maximal Koopman-Invariant Subspace).*** *Let $\mathcal{I}_{\max}$ denote the maximal Koopman-invariant subspace in* $\operatorname{span}(D)$ *and let $C_{\max}$ be a full-column rank matrix such that $D(\cdot)C_{\max}$ spans $\mathcal{I}_{\max}$ (if $\mathcal{I}_{\max} = \{0\}$, we set $C_{\max} = 0$). Then, for any $\epsilon \in [0, 1]$,*

$$\mathcal{R}(C_{\max}) \subseteq \mathcal{R}(C_i), \quad \forall i \in \{0, \ldots, L\},$$

where L and C_i denote, respectively, the termination step and the ith internal matrix of T-SSD.

Proof. The result holds trivially if $\mathcal{I}_{\max} = \{0\}$. For the case $\mathcal{I}_{\max} \neq \{0\}$, we reason by induction. For $i = 0$, columns of C_0 span the whole space. Hence, $\mathcal{R}(C_{\max}) \subseteq \mathcal{R}(C_0)$. Next, assume $\mathcal{R}(C_{\max}) \subseteq \mathcal{R}(C_i)$ for $i \in \{0, 1, \dots, L-1\}$ and let us prove $\mathcal{R}(C_{\max}) \subseteq \mathcal{R}(C_{i+1})$. The invariance of $\mathcal{I}_{\max}$ implies that $\mathcal{R}(D(X)C_{\max}) = \mathcal{R}(D(Y)C_{\max})$. Using the definition of matrices A_0, B_0 in Algorithm 6, this can be equivalently written as $\mathcal{R}(A_0 C_{\max}) = \mathcal{R}(B_0 C_{\max})$. Since $\mathcal{R}(C_{\max}) \subseteq \mathcal{R}(C_i)$, using Lemma 3.7.2, we deduce

$$\mathcal{R}(A_0 C_{\max}) \subseteq \mathcal{R}(A_0 C_i), \quad \mathcal{R}(B_0 C_{\max}) \subseteq \mathcal{R}(B_0 C_i).$$

Hence, $\mathcal{P}_{A_0 C_i} w = w = \mathcal{P}_{B_0 C_i} w$, for all $w \in \mathcal{R}(A_0 C_{\max}) = \mathcal{R}(B_0 C_{\max})$, or equivalently,

$$\|\mathcal{P}_{A_0 C_i} w - \mathcal{P}_{B_0 C_i} w\|_2 = 0, \ \forall w \in \mathcal{R}(A_0 C_{\max}) = \mathcal{R}(B_0 C_{\max}). \tag{5.23}$$

Now, noting that $A_i = A_0 C_i$ and $B_i = B_0 C_i$, one can use Step 6 of Algorithm 6 and write $G_{i+1} v = \mathcal{P}_{A_0 C_i} v - \mathcal{P}_{B_0 C_i} v$, for all $v \in \mathbb{R}^N$. This, combined with (5.23), yields $\mathcal{R}(A_0 C_{\max}) = \mathcal{R}(B_0 C_{\max}) \subseteq \text{null}(G_{i+1})$. Therefore, since the eigenvectors of G_{i+1} with zero eigenvalue span $\text{null}(G_{i+1})$, we deduce from Step 7,

$$\mathcal{R}(A_0 C_{\max}) = \mathcal{R}(B_0 C_{\max}) \subseteq \mathcal{R}(V_{i+1}). \tag{5.24}$$

Based on the induction hypothesis $\mathcal{R}(C_{\max}) \subset \mathcal{R}(C_i)$, and noting that $C_{\max}$ and C_i have full column rank ($C_{\max}$ by definition and C_i from Lemma 5.3.4(ii)), there exits a full-column rank matrix F_i such that

$$C_{\max} = C_i F_i. \tag{5.25}$$

Now, using (5.24)-(5.25), in conjunction with Proposition 5.3.2(iii), we deduce $\mathcal{R}(F_i) \subseteq \mathcal{R}(E_{i+1})$. Consequently, one can use Lemma 3.7.2 and write $\mathcal{R}(C_{\max}) = \mathcal{R}(C_i F_i) \subseteq \mathcal{R}(C_i E_{i+1}) = \mathcal{R}(C_{i+1})$, concluding the proof. $\square$

Theorem 5.4.4 implies that the subspace identified by T-SSD contains the maximal Koopman-invariant subspace in $\operatorname{span}(D)$.

Corollary 5.4.5. ***(T-SSD Subspace Contains the Maximal Koopman-Invariant Subspace).*** *Let $\mathcal{I}_{\max}$ be the maximal Koopman-invariant subspace in $\operatorname{span}(D)$. Given $\epsilon \in [0,1]$, let $C_{\text{T-SSD}}$ and $D_{\text{T-SSD}}$ be the output and dictionary identified by T-SSD according to (5.13)-(5.14). Then, $\mathcal{I}_{\max} \subseteq \operatorname{span}(D_{\text{T-SSD}})$.*

The next result shows that the eigendecomposition of $K_{\text{T-SSD}}$ captures all Koopman eigenfunctions (and corresponding eigenvalues) in the span of the original dictionary.

Proposition 5.4.6. ***($K_{\text{T-SSD}}$ Captures All Koopman Eigenfunctions in $\operatorname{span}(D)$).*** *Let ϕ be a Koopman eigenfunction in $\operatorname{span}(D)$ with eigenvalue λ. For $\epsilon \in [0,1]$, let $K_{\text{T-SSD}}$ in (5.15) be the T-SSD predictor matrix. Then, $\phi \in \operatorname{span}(D_{\text{T-SSD}})$ and there exists w with $K_{\text{T-SSD}} w = \lambda w$ such that $\phi(\cdot) = D_{\text{T-SSD}}(\cdot) w$.*

Proof. Note that ϕ must belong to the maximal Koopman-invariant subspace $\mathcal{I}_{\max}$ in $\operatorname{span}(D)$ which, from Corollary 5.4.5, is included in $\operatorname{span}(D_{\text{T-SSD}}) = \operatorname{span}(D(\cdot) C_{\text{T-SSD}})$. Therefore, there exists a complex vector w of appropriate size such that $\phi(\cdot) = D_{\text{T-SSD}}(\cdot) w$. Using now the interpretation of $K_{\text{T-SSD}}$ as the EDMD solution with dictionary $D_{\text{T-SSD}}$ and data X, Y, it follows from Lemma 3.2.1(ii) that $K_{\text{T-SSD}} w = \lambda w$, as claimed. $\square$

Proposition 5.4.6 states that all eigenfunctions in the span of the original dictionary D belong to the set of approximated eigenfunctions calculated with the dictionary $D_{\text{T-SSD}}$ defined by T-SSD.

Remark 5.4.7. ***(Monotonicity of T-SSD Subspaces).*** In general, the output of the T-SSD algorithm is not monotonic as a function of the design parameter ϵ, i.e., it might be the case that $\operatorname{span}(D^{\epsilon_1}_{\text{T-SSD}}) \not\subset \operatorname{span}(D^{\epsilon_2}_{\text{T-SSD}})$ for $\epsilon_1 < \epsilon_2$. In case monotonicity is desirable for a specific application, one can modify Step 7 of Algorithm 6 to remove only the eigenvector with the largest eigenvalue (in magnitude) that exceeds the desired accuracy level. This modification ensures monotonicity in ϵ at the cost of requiring the modified algorithm more iterations to terminate. All the results remain valid for the modified version of the algorithm. $\square$

5.5 EDMD and SSD are Special Cases of T-SSD

Consistent with our assertion that T-SSD balances accuracy and expressiveness, here we show that EDMD on the original dictionary (maximum expressiveness) corresponds to T-SSD with $\epsilon = 1$ and that SSD (maximum accuracy) corresponds to T-SSD with $\epsilon = 0$[4]. We start by showing an important property of EDMD.

Lemma 5.5.1. ***(Linear Transformations Do not Change the Information Extracted by EDMD).*** *Let D_1 and D_2 be two dictionaries such that $D_1(\cdot) = D_2(\cdot)R$, with R invertible. Let Assumption 4.1.1 hold for both dictionaries given data matrices X and Y.*

[4]We refer to K_{SSD} and $K_{\text{T-SSD}}$ as SSD and T-SSD Koopman approximations, which can be calculated by applying EDMD on dictionaries identified by SSD and T-SSD respectively.

Define

$$K^1_{\text{EDMD}} = \text{EDMD}(D_1, X, Y) = D_1(X)^\dagger D_1(Y),$$

$$K^2_{\text{EDMD}} = \text{EDMD}(D_2, X, Y) = D_2(X)^\dagger D_2(Y).$$

Then, $K^1_{\text{EDMD}} = R^{-1} K^2_{\text{EDMD}} R$. *Therefore,* (λ, v) *is an eigenpair of* K^1_{EDMD} *if and only if* (λ, Rv) *is an eigenpair of* K^2_{EDMD}.

Proof. Based on Assumption 4.1.1, we have $K^1_{\text{EDMD}} = \big(D_1(X)^T D_1(X)\big)^{-1} D_1(X)^T D_1(Y)$. Using $D_1(\cdot) = D_2(\cdot)R$,

$$\begin{aligned} K^1_{\text{EDMD}} &= \big(R^T D_2(X)^T D_2(X) R\big)^{-1} R^T D_2(X)^T D_2(Y) R \\ &= R^{-1}\big(D_2(X)^T D_2(X)\big)^{-1} D_2(X)^T D_2(Y) R \\ &= R^{-1} D_2(X)^\dagger D_2(Y) R = R^{-1} K^2_{\text{EDMD}} R. \end{aligned}$$

The rest follows from the properties of similarity transformations. □

Lemma 5.5.1 states that the dynamical information captured by the EDMD algorithm remains the same under linear transformation of the dictionary. Note that the result does not require the dictionaries to span a Koopman-invariant subspace. We are ready to show that EDMD applied to the original dictionary is a special case of T-SSD.

Theorem 5.5.2. ***(EDMD is a Special Case of T-SSD with*** $\epsilon = 1$***).*** *For* $\epsilon = 1$, *let* $D_{\text{T-SSD}}$ *be the dictionary identified by T-SSD, cf.* (5.14). *Then,* $\text{span}(D_{\text{T-SSD}}) = \text{span}(D)$, *and* $K_{\text{T-SSD}} = \text{EDMD}(D_{\text{T-SSD}}, X, Y)$ *and* $K_{\text{EDMD}} = \text{EDMD}(D, X, Y)$ *are similar and capture* *the same dynamical information.*

Proof. In the first iteration of Algorithm 6, one can use Step 6 and the definition of A_0 and B_0 to write

$$G_1 = A_0 A_0^\dagger - B_0 B_0^\dagger = \mathcal{P}_{D(X)} - \mathcal{P}_{D(Y)}.$$

Since G_1 is symmetric, its eigenvalues are real. Moreover, they belong to $[-1, 1]$, see e.g. [AHT85, Lemma 1]. Therefore, since $\epsilon = 1$, using Step 7, one can deduce that the columns of V_1 span $\mathbb{R}^N$. As a result,

$$\mathcal{R}(D(X)) = \mathcal{R}(A_0) = \mathcal{R}(A_0 I_{N_d}) \subseteq \mathcal{R}(V_1) = \mathbb{R}^N,$$
$$\mathcal{R}(D(Y)) = \mathcal{R}(B_0) = \mathcal{R}(B_0 I_{N_d}) \subseteq \mathcal{R}(V_1) = \mathbb{R}^N.$$

This, combined with the maximality of E_1 defined in Step 8, cf. Proposition 5.3.2(iii), implies $\mathcal{R}(I_{N_d}) \subseteq \mathcal{R}(E_1)$. Hence, E_1 is nonzero and has full column rank (cf. Proposition 5.3.2(i)). As a result, nothing that $\sharp\text{rows}(E_1) = N_d$, we deduce that E_1 is a nonsingular square matrix. Therefore, $\mathcal{R}(C_1) = \mathcal{R}(C_0 E_1) = \mathbb{R}^{N_d}$. This and the fact that E_1 is square mean that the condition in Step 15 is met and the algorithm executes Steps 16-17. Consequently, $C_{\text{T-SSD}} = C_1$ is a nonsingular square matrix and $\text{span}(D_{\text{T-SSD}}(\cdot)) = \text{span}(D(\cdot)C_{\text{T-SSD}}) = \text{span}(D(\cdot))$, so $D_{\text{T-SSD}}$ is a (potentially different) basis for the space spanned by D. The rest of the statement follows from Lemma 5.5.1. □

The SSD algorithm is also a special case of T-SSD.

Theorem 5.5.3. ***(SSD is a Special Case of T-SSD with*** $\epsilon = 0$***).*** *Let* $D_{\text{SSD}} = \text{SSD}(D(X), D(Y))$ *be the dictionary identified by SSD algorithm (cf. Algorithm 1), and, for* $\epsilon = 0$*, let* $D_{\text{T-SSD}}$ *be the dictionary identified by T-SSD, cf.* (5.14)*. Then,* $\text{span}(D_{\text{T-SSD}}) = \text{span}(D_{\text{SSD}})$*, and* $K_{\text{T-SSD}} = \text{EDMD}(D_{\text{T-SSD}}, X, Y)$ *and* $K_{\text{SSD}} = \text{EDMD}(D_{\text{SSD}}, X, Y)$ *are similar and capture the same dynamical information.*

Proof. Since $\epsilon = 0$, Theorem 5.4.1 implies that $\mathcal{R}(D(X)C_{\text{T-SSD}})$ and $\mathcal{R}(D(Y)C_{\text{T-SSD}})$ are 0-apart. Therefore, from Lemma 5.2.2, $\mathcal{R}(D(X)C_{\text{T-SSD}}) = \mathcal{R}(D(X)C_{\text{T-SSD}})$. This, together with Theorem 3.3.1(iii), implies

$$\mathcal{R}(C_{\text{T-SSD}}) \subseteq \mathcal{R}(C_{\text{SSD}}). \tag{5.26}$$

If $C_{\text{SSD}} = 0$, then $C_{\text{T-SSD}} = 0$, and the proof is complete. Suppose instead that $C_{\text{SSD}} \neq 0$, with full column rank, cf. Theorem 3.3.1(i). We use induction to prove that $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_i)$, where C_i is the internal matrix of the T-SSD algorithm for $i \in \{0, \dots, L\}$ and L is the iteration at which it terminates. When $i = 0$, the columns of $C_0 = I_{N_d}$ span $\mathbb{R}^{N_d}$ and, therefore, $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_0)$. Assume then that $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_i)$ for $i \in \{0, \dots, L-1\}$, and let us prove that $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_{i+1})$.

Based on Theorem 3.3.1(ii), we have $\mathcal{R}(D(X)C_{\text{SSD}}) = \mathcal{R}(D(Y)C_{\text{SSD}})$. This, together the definition of matrices A_0, B_0 in Algorithm 6 and the fact that $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_i)$, yields

$$\mathcal{P}_{A_0 C_i} w = \mathcal{P}_{B_0 C_i} w = w, \tag{5.27}$$

for all $w \in \mathcal{R}(A_0 C_{\text{SSD}}) = \mathcal{R}(B_0 C_{\text{SSD}})$. Now, since $A_i = A_0 C_i$ and $B_i = B_0 C_i$ at iteration

$i+1$ of the T-SSD algorithm, $G_{i+1}v = \mathcal{P}_{A_0C_i}v - \mathcal{P}_{B_0C_i}v$, for all $v \in \mathbb{R}^N$. This, together with (5.27), implies that $\mathcal{R}(A_0C_{\text{SSD}}) = \mathcal{R}(B_0C_{\text{SSD}}) \subseteq \text{null}(G_{i+1})$. Since $\epsilon = 0$, from Step 7 we know that V_{i+1} is a basis for $\text{null}(G_{i+1})$, and therefore

$$\mathcal{R}(A_0C_{\text{SSD}}) = \mathcal{R}(B_0C_{\text{SSD}}) \subseteq \mathcal{R}(V_{i+1}). \tag{5.28}$$

By the induction hypothesis $\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_i)$. This, together with the fact that C_{SSD} and C_i have full column rank (the latter because of Lemma 5.3.4(ii)), implies that there exists a matrix F_i with full column rank such that

$$C_{\text{SSD}} = C_iF_i. \tag{5.29}$$

Using now (5.28)-(5.29) together with the fact that $A_i = A_0C_i$, $B_i = B_0C_i$, one can invoke Proposition 5.3.2(iii) to deduce that $\mathcal{R}(F_i) \subseteq \mathcal{R}(E_{i+1})$. Consequently,

$$\mathcal{R}(C_{\text{SSD}}) = \mathcal{R}(C_iF_i) \subseteq \mathcal{R}(C_iE_{i+1}) = \mathcal{R}(C_{i+1}).$$

Hence, the induction is complete and

$$\mathcal{R}(C_{\text{SSD}}) \subseteq \mathcal{R}(C_i), \forall i \in \{1, \dots, L\}. \tag{5.30}$$

Since C_{SSD} is nonzero and has full column rank, one can deduce that C_L is nonzero and has full column rank as a result of Lemma 5.3.4(iii). Consequently, the T-SSD algorithm terminates by executing Steps 16-17. Therefore, $C_{\text{T-SSD}} = C_L$ and using (5.26) and (5.30), we have $\mathcal{R}(C_{\text{T-SSD}}) = \mathcal{R}(C_{\text{SSD}})$ and consequently $\text{span}(D_{\text{T-SSD}}) = \text{span}(D_{\text{SSD}})$. The rest of the statement follows from Lemma 5.5.1. $\square$

It is worth mentioning that, when implementing T-SSD for $\epsilon = 0$, we have found it useful to set ϵ to be a small positive number (instead of zero) to avoid complications by round-off errors. Given Theorem 5.5.3, the subspace identified by T-SSD for $\epsilon = 0$ enjoys important dynamical properties, cf. Chapter 3: under reasonable conditions on data sampling, the identified subspace is the maximal Koopman-invariant subspace in the span of the dictionary almost surely. Moreover, the eigenfunctions and predictors identified by T-SSD are almost surely exact.

5.6 Efficient Implementation of T-SSD

Here, we propose a modification to the implementation of the T-SSD algorithm on digital computers to increase efficiency. This is based on the following observation: a close look at the form of the matrix $G_i \in \mathbb{R}^{N \times N}$ in Step 6 of Algorithm 6 as a difference of projections reveals that its eigenvectors are either in or orthogonal to the subspace $\mathcal{R}(A_{i-1}) + \mathcal{R}(B_{i-1})$, see e.g., [AHT85]. However, in Step 8, the matrix E_i satisfies $\mathcal{R}(A_{i-1}E_i), \mathcal{R}(B_{i-1}E_i) \subseteq \mathcal{R}(V_i)$. Hence, the Symmetric-Intersection function filters out all eigenvectors of G_i that are orthogonal to $\mathcal{R}(A_{i-1}) + \mathcal{R}(B_{i-1})$, i.e., these eigenvectors are never used. This is despite the fact that, since generally $N \gg N_d$, such eigenvectors form a majority of eigenvectors of G_i (at least $N - 2N_d$ out of N).

This motivates us to seek a method that bypasses the calculation of the unused eigenvectors of G_i. To achieve this goal, let H_i be a matrix such that

$$\mathcal{R}(H_i) := \mathcal{R}([A_{i-1}, B_{i-1}]), \quad H_i^T H_i = I_{\sharp\mathrm{cols}(H_i)}. \tag{5.31}$$

The columns of H_i form an orthonormal basis of $\mathcal{R}(A_{i-1})+\mathcal{R}(B_{i-1})$. One can calculate H_i by applying the Gram–Schmidt process, or other closely related orthogonal factorization method such as QR decomposition (see e.g. [TB97]), on $[A_{i-1}, B_{i-1}]$. The next result shows that the eigendecomposition of the matrix $H_i^T G_i H_i$ completely captures the eigendecomposition of G_i on $\mathcal{R}(A_{i-1}) + \mathcal{R}(B_{i-1})$.

Proposition 5.6.1. ***(Eigenvectors of*** $H_i^T G_i H_i$ ***Characterize All Eigenvectors of*** G_i ***in*** $\mathcal{R}(A_{i-1})+\mathcal{R}(B_{i-1})$***).*** *Let* G_i *as defined in Step 6 of Algorithm 6, and let* H_i *satisfy* (5.31). *Then,* $w \in \mathbb{C}^{N_d} \setminus \{0\}$ *is an eigenvector of* $H_i^T G_i H_i$ *with eigenvalue* λ *if and only if* $v = H_i w$ *is an eigenvector of* G_i *with eigenvalue* λ.

Proof. ($\Leftarrow$) By hypothesis, $G_i H_i w = \lambda H_i w$. Hence, $H_i^T G_i H_i w = \lambda H_i^T H_i w = \lambda w$ (where we have used (5.31)).

($\Rightarrow$) By hypothesis, $H_i^T G_i H_i w = \lambda w$. Using (5.31), this can be rewritten as

$$H_i^T (G_i H_i w - \lambda H_i w) = 0. \tag{5.32}$$

By definition of G_i, we can write $G_i H_i w = \mathcal{P}_{A_{i-1}}(H_i w) - \mathcal{P}_{B_{i-1}}(H_i w)$. From (5.31), we have $\mathcal{R}(A_{i-1}), \mathcal{R}(B_{i-1}) \subseteq \mathcal{R}(H_i)$. Since $\mathcal{P}_{A_{i-1}}(H_i w) \in \mathcal{R}(A_{i-1})$ and $\mathcal{P}_{B_{i-1}}(H_i w) \in \mathcal{R}(B_{i-1})$, we deduce that $G_i H_i w \in \mathcal{R}(H_i)$, and consequently, $G_i H_i w - \lambda H_i w \in \mathcal{R}(H_i)$. However, from (5.32), $(G_i H_i w - \lambda H_i w) \in \text{null}(H_i^T)$. Therefore, since $\mathcal{R}(H_i) \perp \text{null}(H_i^T)$, we conclude $G_i H_i w - \lambda H_i w = 0$, as claimed. □

Based on Proposition 5.6.1, we modify T-SSD to achieve higher computational efficiency. Formally, the **Efficient T-SSD** algorithm replaces Steps 6 and 8 of Algorithm 6 by

```
6.a:  $H_i \leftarrow \text{basis}([A_i, B_i])$
6.b:  $G_i \leftarrow H_i^T (A_{i-1} A_{i-1}^\dagger - B_{i-1} B_{i-1}^\dagger) H_i$
  8:  $E_i \leftarrow \text{Symmetric-Intersection}(H_i V_i, A_{i-1}, B_{i-1})$
```

These steps bypass the computation of the (unused) eigenvectors of G_i that are orthogonal to $\mathcal{R}(A_{i-1}) + \mathcal{R}(B_{i-1})$ in the original T-SSD implementation.

Remark 5.6.2. ***(Computational Complexity of Efficient T-SSD).*** Given N data snapshots and N_d dictionary functions, and considering the complexity of scalar operations as $O(1)$, the most time-consuming steps of Efficient T-SSD are calculating H_i in (5.31) and the null space calculations in the function Symmetric-Intersection, which can be done in $O(N N_d^2)$ operations. Since the algorithm terminates after at most N_d iterations, cf. Proposition 5.3.3, the overall complexity is $O(N N_d^3)$. Compared to T-SSD, cf. Remark 5.3.5, the efficient T-SSD algorithm provides a reduction of $O(N^2 N_d^{-2})$, leading to a drastic reduction in run time for typical situations, where $N \gg N_d$. □

5.7 Simulation Results

Here, we illustrate the effectiveness of our proposed methods using three examples.

Example 5.7.1. ***(Hopf Normal Form).*** Consider the system [BPK16, MM12] on $\mathcal{M} = [-2, 2]^2$,

$$\dot{x_1} = x_1 + 2x_2 - x_1(x_1^2 + x_2^2),$$
$$\dot{x_2} = -2x_1 + x_2 - x_2(x_1^2 + x_2^2), \tag{5.33}$$

with state $x = [x_1, x_2]^T$, which admits an attractive periodic orbit. We consider the discretized version of (5.33) with time step $\Delta t = 0.01s$ and gather $N = 10^4$ data snapshots in matrices X and Y, with initial conditions uniformly selected from $\mathcal{M}$. We consider the space of all polynomials up to degree 10 spanned by all the $N_d = 66$ distinct monomials in the form of $\prod_{i=1}^{10} \alpha_i$, with $\alpha_i \in \{1, x_1, x_2\}$ for $i \in \{1, \ldots, 10\}$. To ensure resilience against machine precision errors and providing informative representations, we choose a dictionary D with $N_d = 66$ functions such that the columns of $D(X)$ are orthonormal[5].

We implement the Efficient T-SSD algorithm, cf. Section 5.6, with $\epsilon \in \{0.02, 0.05, 0.1, 0.15, 0.2\}$. Table 5.1 shows the dimension of the identified dictionary, $D_{\text{T-SSD}}$, versus the value of the design parameter ϵ. For $\epsilon = 0.2$, T-SSD identifies the original dictionary, certifying that the range spaces of $D(X)$ and $D(Y)$ are 0.2-apart. On the other hand, the one-dimensional subspace identified by $\epsilon = 0.02$ is in fact the maximal Koopman-

[5]This dictionary can be found by first forming a dictionary comprised of the monomials and then performing a linear transformation on the dictionary to make the columns orthonormal. The linear transformation does not impact the captured dynamical information (cf. Lemma 5.5.1).

invariant subspace of span(D), spanned by the trivial eigenfunction $\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$.

Table 5.1: Dimension of subspace identified by Efficient T-SSD vs ϵ for (5.33).

$\boldsymbol{\epsilon}$	0.02	0.05	0.10	0.15	0.20
dim $\boldsymbol{D}_{\textbf{T-SSD}}$	1	6	8	16	66

To demonstrate the effectiveness of the T-SSD algorithm in approximating Koopman eigenfunctions and invariant subspaces, we focus on the subspace identified with $\epsilon = 0.05$. In accordance with Proposition 5.4.6, T-SSD identifies the trivial eigenfunction $\phi(x) \equiv 1$ spanning the maximal Koopman-invariant subspace of span(D). T-SSD also approximates another real-valued eigenfunction with eigenvalue $\lambda = 0.9066$, whose absolute value is illustrated in Figure 5.1(right). Given that $|0.9066| < 1$, this eigenfunction predicts the existence of a forward invariant set (the periodic orbit in Figure 5.1(left)) at its zero-level set.

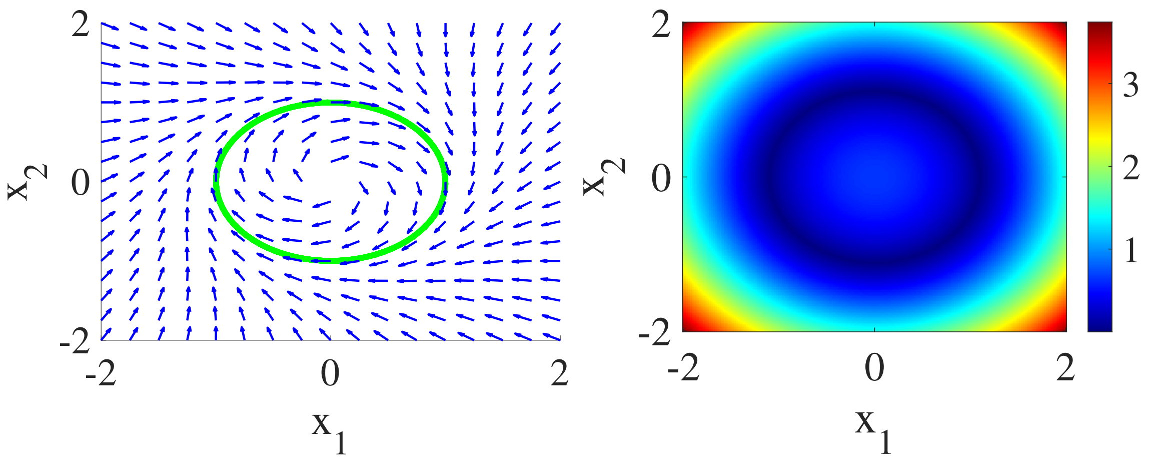

Figure 5.1: Vector field and limit cycle of system (5.33) (left) and the absolute value of eigenfunction with eigenvalue $\lambda = 0.9066$ (right).

In addition, T-SSD also identifies two pairs of complex eigenfunctions. For space reasons, we only show in Figure 5.2 one eigenfunction with eigenvalue $\lambda = 0.9938 + 0.0195j$

(the one closest to the unit circle). Its phase characterizes the oscillation of the trajectories in the state space.

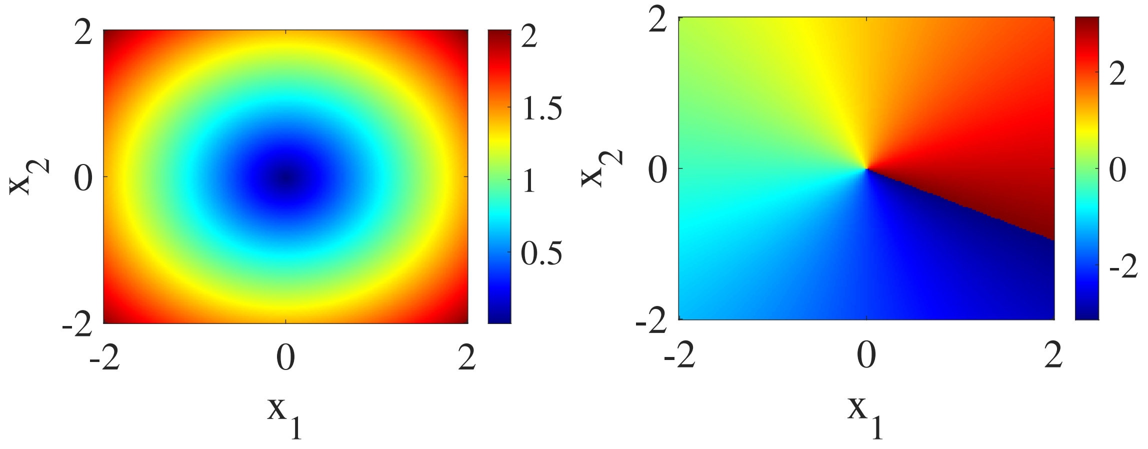

Figure 5.2: Absolute value (left) and phase (right) of the eigenfunction with eigenvalue $\lambda = 0.9938 + 0.0195j$ for (5.33).

To illustrate the efficacy of our algorithm regarding the prediction accuracy of the dictionary, we consider the relative linear prediction error associated with a dictionary $\mathcal{D}$ at point x given data snapshot matrices X and Y defined by

$$E_{\text{relative}}(x) := \frac{\|\mathcal{D} \circ T(x) - \mathcal{D}(x)K\|_2}{\|\mathcal{D} \circ T(x)\|_2} \times 100, \tag{5.34}$$

where $K = \text{EDMD}(\mathcal{D}, X, Y)$. Figure 5.3 compares this error on the state space $\mathcal{M}$ for the dictionary $D_{\text{T-SSD}}$ identified by T-SSD with $\epsilon = 0.05$ and for the original dictionary D. This error is evaluated at points other than the training data X and clearly shows the advantage of $D_{\text{T-SSD}}$ over the original dictionary both in prediction errors and capturing the radial symmetry of the vector field.

Noting that the error in (5.34) depends on the dictionary and does not provide information about the subspace it spans (or the individual members of the subspace), we also

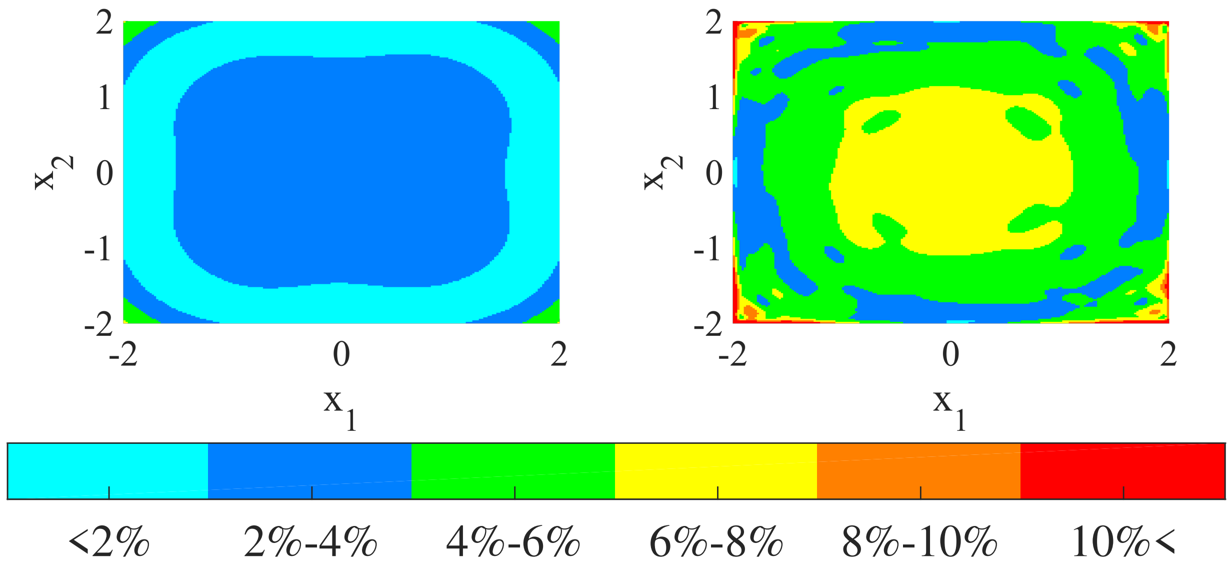

Figure 5.3: Relative linear prediction error for dictionary identified by T-SSD ($\epsilon = 0.05$) (left) and the original dictionary (right) for (5.33).

consider the latter. For this reason, we use the data sampling strategy used earlier to build a test data set denoted by snapshot matrices X_{test} and Y_{test} with $N_{\text{test}} = 10^4$ samples. Given a dictionary $\mathcal{D}$, we evaluate the invariance proximity of span($\mathcal{D}$) as the smallest ϵ such that $\mathcal{R}(\mathcal{D}(X_{\text{test}}))$ and $\mathcal{R}(\mathcal{D}(Y_{\text{test}}))$ are ϵ-apart. This data-driven measure is equivalent to the maximum relative root mean square error for a function in the span of a given dictionary $\mathcal{D}$ on the test data defined as

$$\text{RRMSE}_{\max}(\mathcal{D}, X_{\text{test}}, Y_{\text{test}}) = \max_{f \in \text{span}(\mathcal{D})} \frac{\sqrt{\frac{1}{N_t} \sum_{i=1}^{N_t} |\mathcal{K}f(x_i) - \mathfrak{P}^{\text{T-SSD}}_{\mathcal{K}f}|^2}}{\sqrt{\frac{1}{N_t} \sum_{i=1}^{N_t} |\mathcal{K}f(x_i)|^2}}, \tag{5.35}$$

where x_i^T and y_i^T correspond to the ith rows of X_{test} and Y_{test} respectively, $y_i = T(x_i)$, and T is the map defining the dynamics (5.33). The predictor $\mathfrak{P}^{\text{T-SSD}}_{\mathcal{K}f}$ is defined in (5.16) and calculated based on the test data. It is important to note that the evaluation of the error in (5.35) goes beyond the assumptions of Theorem 5.4.2, since the dictionary is identified with the original data X, Y, but the error is evaluated on the test data $X_{\text{test}}, Y_{\text{test}}$ (instead, the guarantee of Theorem 5.4.2 are only valid when the error is evaluated on the original data). Following the reasoning in the proof of Theorem 5.4.2 and Definition 5.2.1, one can analytically show that

$$\text{RRMSE}_{\max}(\mathcal{D}, X_{\text{test}}, Y_{\text{test}}) = \lambda_{\max}(\mathcal{P}_{\mathcal{D}(X_{\text{test}})} - \mathcal{P}_{\mathcal{D}(Y_{\text{test}})}),$$

where $\lambda_{\max}$ denotes the largest eigenvalue of the argument. Table 5.2 shows the maximum relative root mean square error for the subspaces identified by T-SSD given different values of ϵ. According to Table 5.2, despite the fact that we have used different data for identification and evaluation, the error on the test data satisfies the upper bound accuracy requirement enforced by the accuracy parameter ϵ .

Table 5.2: Maximum Relative Root Mean Square Error vs ϵ for (5.33).

$\boldsymbol{\epsilon}$	0.02	0.05	0.10	0.15	0.20
$\text{RRMSE}_{\max}$	~ 0	0.037	0.100	0.115	0.185

Example 5.7.2. ***(Duffing System).*** Consider the Duffing system [WKR15] on $\mathcal{M} = [-2, 2]^2$,

$$\dot{x_1} = x_2,$$
$$\dot{x_2} = -0.5x_2 + x_1(1 - x_1^2), \tag{5.36}$$

with state $x = [x_1, x_2]^T$, which has an unstable equilibrium at the origin and two asymptotically stable equilibria at $(-1, 0)$ and $(1, 0)$. We consider the discretized version of (5.36) with time step $\Delta t = 0.02s$ and gather $N = 10^4$ data snapshots in matrices X and Y from 5000 trajectories with length equal to two time steps and initial conditions uniformly selected from $\mathcal{M}$. Similarly to the previous example, we use a dictionary D with $N_d = 66$ elements spanning the space of all polynomials up to degree 10 such that the columns of $D(X)$ are orthonormal.

We apply the Efficient T-SSD algorithm, cf. Section 5.6, with $\epsilon \in \{0.01, 0.02, 0.08, 0.14, 0.2, 0.26\}$. Table 5.3 shows the dimension of the identified dictionary, $D_{\text{T-SSD}}$, versus the value of the design parameter ϵ. For $\epsilon = 0.26$, T-SSD identifies the original dictionary, certifying that the range spaces of $D(X)$ and $D(Y)$ are 0.26-apart. On the other hand, the one-dimensional subspace identified by $\epsilon = 0.01$ is in fact the maximal Koopman-invariant subspace of span(D), spanned by the trivial eigenfunction $\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$.

Table 5.3: Dimension of subspace identified by Efficient T-SSD vs ϵ for (5.36).

$\boldsymbol{\epsilon}$	0.01	0.02	0.08	0.14	0.20	0.26
dim $\boldsymbol{D}_{\textbf{T-SSD}}$	1	2	20	44	58	66

To demonstrate the effectiveness of the T-SSD algorithm in approximating Koopman eigenfunctions and invariant subspaces, we focus on the subspace identified with $\epsilon = 0.02$. Consistent with Proposition 5.4.6, T-SSD identifies the trivial eigenfunction $\phi(x) \equiv 1$ spanning the maximal Koopman-invariant subspace of span(D). T-SSD also approximates another real-valued eigenfunction with eigenvalue $\lambda = 0.9839$ depicted in Figure 5.4(right), which clearly captures the attractiveness of the asymptotically stable equilibria and the general behavior of the vector field depicted in Figure 5.4(left).

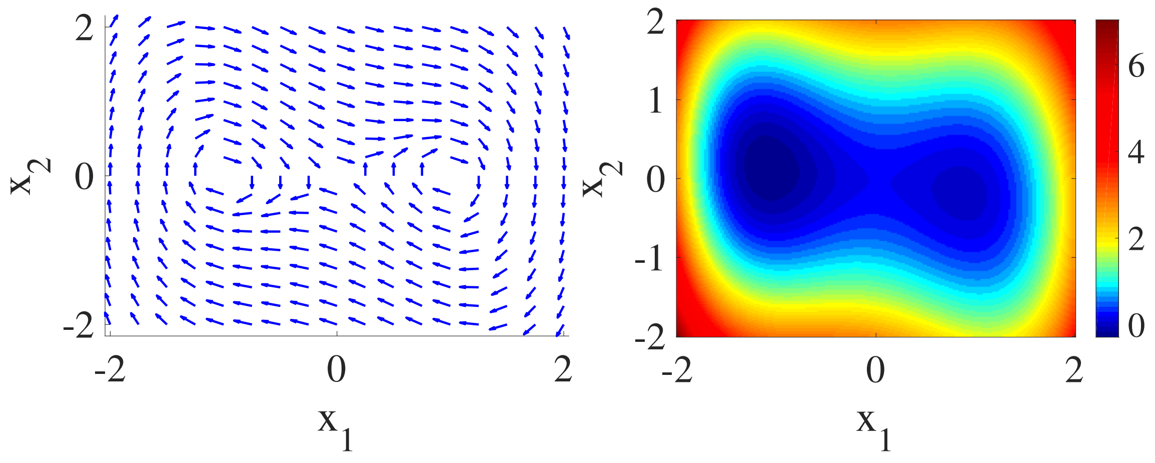

Figure 5.4: Vector field (left) and eigenfunction with eigenvalue $\lambda = 0.9839$ (right) for (5.36).

To illustrate the efficacy of our algorithm regarding the prediction accuracy of the dictionary, Figure 5.5 compares the relative linear prediction error (5.34) on the state space $\mathcal{M}$ for the dictionary $D_{\text{T-SSD}}$ identified by T-SSD with $\epsilon = 0.02$ and for the original dictionary D evaluated at out-of-sample points other than X. Figure 5.5 clearly shows the effectiveness of the T-SSD algorithm in improving the prediction accuracy.

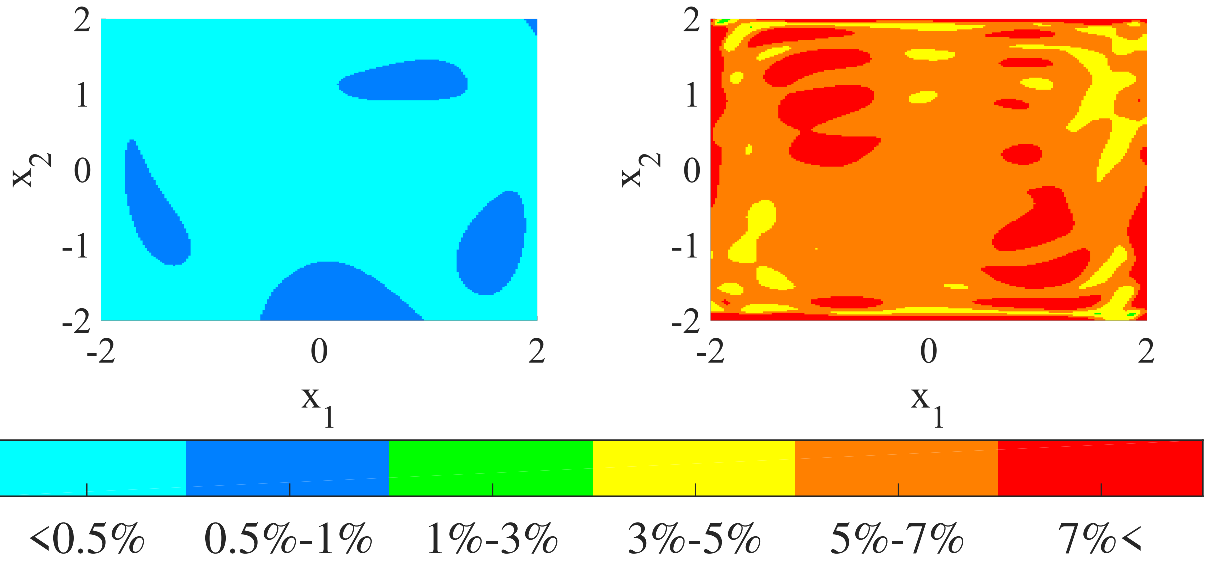

Figure 5.5: Relative linear prediction error for dictionary identified by T-SSD ($\epsilon = 0.02$) (left) and the original dictionary (right) for (5.36).

To analyze the error for individual functions in the identified subspaces by T-SSD, we form a random test data set X_{test}, Y_{test} gathered with the same number of elements and sampling strategy used for X and Y. Table 5.4 shows the maximum relative root mean square error defined in (5.35) for the subspaces identified by T-SSD given different values of ϵ. Despite the fact that we use different data for identification and evaluation, Table 5.4 shows the effectiveness of the T-SSD algorithm for identifying subspaces on which all functions have prediction errors characterized by the accuracy parameter ϵ.

Table 5.4: Maximum Relative Root Mean Square Error vs ϵ for (5.36).

$\boldsymbol{\epsilon}$	0.01	0.02	0.08	0.14	0.20	0.26
$\text{RRMSE}_{\max}$	~ 0	0.004	0.054	0.123	0.190	0.236

Example 5.7.3. ***(Consensus on Harmonic Mean).*** Given N_a agents with state $x = [x_1, \dots, x_{N_a}]^T$ communicating through a graph with adjacency matrix A, consider the dynamics

$$\dot{x}_i = N_a \, x_i^2 \, \Upsilon(x)^{-2} \sum_{j=1}^{N_a} a_{ij}(x_j - x_i), \; i \in \{1, \dots, N_a\}, \tag{5.37}$$

where a_{ij} is the element of A on row i and column j and $\Upsilon(x)$ is the harmonic mean of the state elements defined as

$$\Upsilon(x) = N_a \Big(\sum_{k=1}^{N_a} x_k^{-1} \Big)^{-1}.$$

For any initial condition x_0, all the state elements converge to the harmonic mean of the initial condition $\Upsilon(x_0)$ [Cor08, Proposition 10], i.e., the agents achieve consensus on $\Upsilon(x_0)$.

For the purpose of this example, we consider $N_a = 5$ agents communicating through an undirected ring graph and states belonging to the state space $\mathcal{M} = [1,5]^5$. We consider the discretized version of (5.37) with time step $\Delta t = 0.01s$ and gather $N = 4 \times 10^4$ data snapshots in matrices X and Y from 2×10^4 trajectories with length equal to two time steps and initial conditions uniformly selected from $\mathcal{M}$. For our dictionary, we consider the space of all polynomials up to degree 6 and choose a dictionary D with $N_d = 462$ functions spanning the space such that the columns of $D(X)$ are orthonormal.

We apply the Efficient T-SSD algorithm, cf. Section 5.6, with $\epsilon \in \{0.05, 0.15, 0.3, 0.55, 0.8\}$. Table 5.5 shows the dimension of the identified dictionary, $D_{\text{T-SSD}}$, versus the value of the design parameter ϵ. For $\epsilon = 0.8$, T-SSD identifies the original subspace, certifying that the range spaces of $D(X)$ and $D(Y)$ are 0.8-apart. On the other hand, the one-dimensional subspace identified by $\epsilon = 0.05$ is in fact the maximal Koopman-invariant subspace of span(D), spanned by the trivial eigenfunction $\phi(x) \equiv 1$ with eigenvalue $\lambda = 1$.

Table 5.5: Dimension of subspace identified by Efficient T-SSD vs ϵ for (5.37).

$\boldsymbol{\epsilon}$	0.05	0.15	0.30	0.55	0.80
dim $\boldsymbol{D_{\text{T-SSD}}}$	1	14	64	272	462

To illustrate the efficacy of T-SSD algorithm regarding prediction accuracy, we first form a test data set comprised of snapshots matrices X_{test} and Y_{test} sampled with the same sampling strategy and number of samples as X and Y. Figure 5.6 provides histogram plots comparing the relative prediction error defined in (5.34) of the dictionary identified by T-SSD with $\epsilon = 0.15$ and the original dictionary applied on the test data. In Figure 5.6 the horizontal axis denotes the prediction error while the vertical axis shows the percentage of test data per interval. Figure 5.6 clearly shows the effectiveness of the T-SSD algorithm in improving the prediction accuracy.

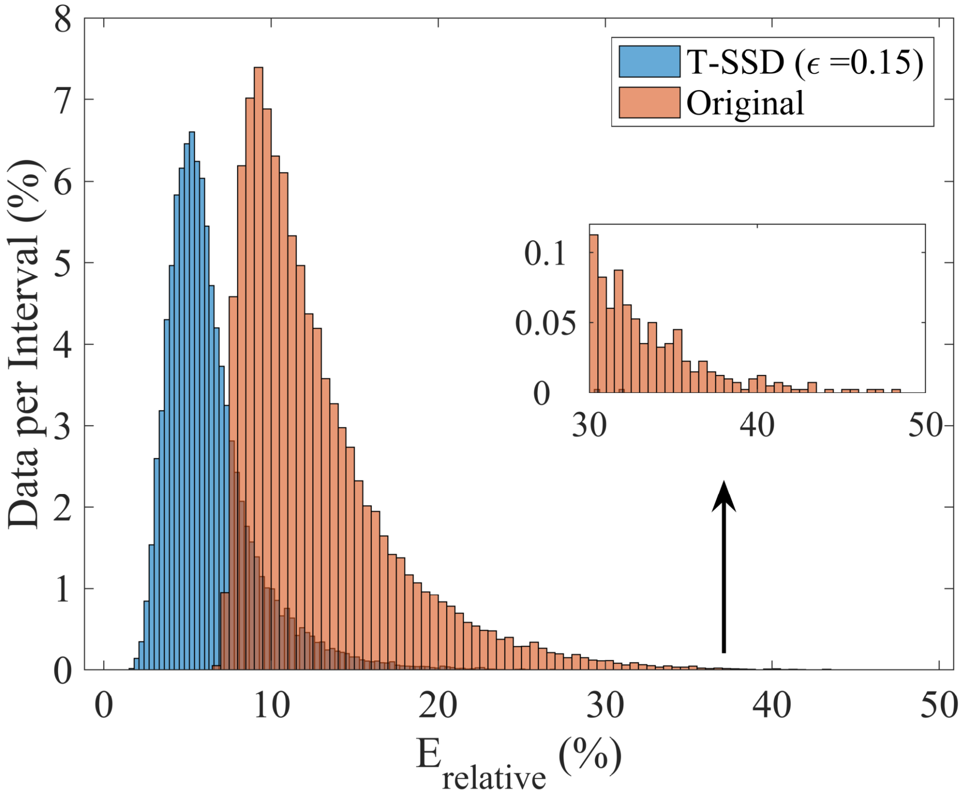

Figure 5.6: Relative linear prediction error on test data for the dictionary identified with T-SSD ($\epsilon = 0.15$) and the original dictionary.

We also consider the prediction accuracy for the individual functions. Table 5.6 shows the maximum relative root mean square error defined in (5.35) for the subspaces identified by T-SSD given different values of ϵ. According to Table 5.6, despite using different data for identification and evaluation, the error on the test data satisfies the upper bound accuracy requirement enforced by the accuracy parameter ϵ.

Table 5.6: Maximum Relative Root Mean Square Error vs ϵ for (5.37).

$\boldsymbol{\epsilon}$	0.05	0.15	0.30	0.55	0.8
$\text{RRMSE}_{\text{max}}$	~ 0	0.144	0.295	0.549	0.769

Acknowledgements

This chapter is taken, in part, from the work [HC21a] which has been submitted for publication as 'Generalizing dynamic mode decomposition: balancing accuracy and expressiveness in Koopman approximations' by M. Haseli and J. Cortés, in Automatica. The dissertation author was the primary investigator and author of this paper. The work in this chapter was partially supported by ONR Award N00014-18-1-2828 and NSF Award IIS-2007141.

Chapter 6

Assessing the Prediction Accuracy and Dictionary Quality of EDMD

In the previous chapters, we observed that the quality of the prediction by Extended Dynamic Mode Decomposition (EDMD) algorithm directly depends on the quality of the particular dictionary's span, specifically on how close it is to being invariant under the Koopman operator. While in the previous chapters, we provided algebraic algorithms to identify Koopman-invariant subspaces, in many applications neural networks and other optimization algorithms are used to find an appropriate subspace. In those methods, the residual error of EDMD is typically used as an objective function. In this chapter, we examine the residual error and show that it does not encode the quality of the function space and is sensitive to the choice of basis. Motivated by this observation, we introduce the novel concept of consistency index. We show that this measure, based on using EDMD forward and backward in time, enjoys a number of desirable qualities that make it suitable for data-driven modeling of dynamical systems: it measures the quality of the function space, it is invariant under the choice of basis, can be computed in closed form from the data, and provides a tight upper-bound for the relative root mean square error of all function predictions on the entire span of the dictionary.

6.1 Motivation: Residual Error of EDMD Does Not Characterize the Dictionary's Quality

As mentioned in Section 2.4 and observed in Chapters 3-5, the quality of the dictionary used for the EDMD method directly impacts its accuracy. Since in general the system is unknown, in many applications, the *residual error* of EDMD, $\|D(Y) - D(X)K_{\text{EDMD}}\|_F$, is commonly used as an objective function in optimization and neural network-based dictionary learning schemes. However, it is important to note that even though a high quality subspace (close to Koopman-invariant) leads to small residual error, the converse is not true, i.e., a dictionary D with small (but nonzero) residual error does not necessarily mean that EDMD's prediction is accurate on $\text{span}(D)$. We illustrate this next.

Example 6.1.1. ***(Residual Error is not Invariant under Linear Transformation of Dictionary).*** Consider the linear system $x^+ = 0.5x$ and the vector space of functions $\mathcal{S} = \text{span}\{x, x^3 - x^2\}$. To apply EDMD, we gather $N = 1000$ data snapshots from trajectories of the system with length of two time steps and initial conditions uniformly selected from $[-2, 2]$, and form data matrices $X, Y \in \mathbb{R}^{N\times 1}$. We consider a family of dictionaries parameterized by $\alpha \in \mathbb{R} \setminus \{0\}$ in the form of

$$D_\alpha(x) = [x, x + \alpha(x^3 - x^2)].$$

Note that each D_α is a basis for $\mathcal{S}$, and all the dictionaries are related by nonsingular linear transformations. We also define two notions of prediction accuracy: the residual error E of EDMD and its normalized version E_{rel},

$$E(\alpha) = \|D_\alpha(Y) - D_\alpha(X)K_\alpha\|_F, \qquad\qquad E_{\text{rel}}(\alpha) = \frac{E(\alpha)}{\|D_\alpha(Y)\|_F},$$

where $K_\alpha = \text{EDMD}(D_\alpha, X, Y)$.

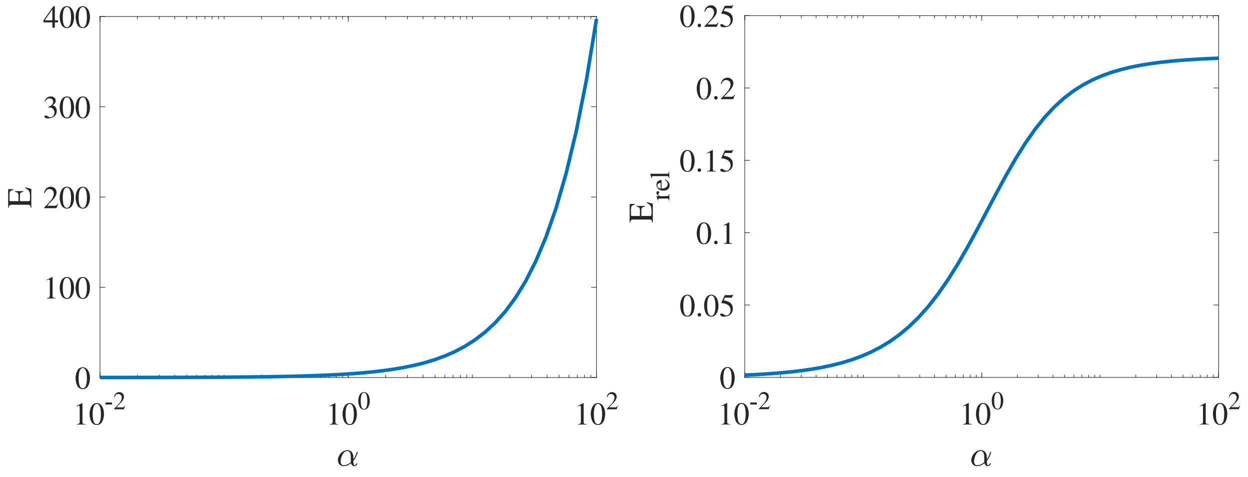

Figure 6.1: Residual error (left) and relative residual error (right) of EDMD for $\alpha \in [0.01, 100]$.

Figure 6.1 shows the aforementioned notions of error versus the value of $\alpha \in [0.01, 100]$. Figure 6.1 clearly demonstrates the sensitivity of errors to the choice of basis for $\mathcal{S}$ despite the invariance of $\text{spec}(K_{\text{EDMD},\alpha})$ under the choice of basis (cf. Lemma 5.5.1). In fact, by tuning the value of α, one can make both errors arbitrarily close to zero. As a result, using the residual error as a measure to assess the quality of the space or as an objective function in optimization or neural network-based dictionary learning schemes can lead to erroneous results. □

The observations in Example 6.1.1 prompts us to search for a better measure of the dictionary's quality and therefore the EDMD's prediction accuracy as we explain next.

6.2 Problem Statement

Consider [1] the following discrete-time system defined over the state space $\mathcal{M} \subseteq \mathbb{R}^n$

$$x^+ = T(x), \tag{6.1}$$

and let $\mathcal{K}$ be its associated Koopman operator defined on the linear space $\mathcal{F}$ comprised of functions mapping elements in $\mathcal{M}$ to $\mathcal{M}$. Moreover, let $X, Y \in \mathbb{R}^{N \times n}$ be matrices comprised of N data snapshots such that

$$y_i = T(x_i),\ i \in \{1, \dots, N\}, \tag{6.2}$$

where x_i^T and y_i^T are ith rows of X and Y. In addition, let $D : \mathcal{M} \to \mathbb{R}^{1 \times N_d}$ be a dictionary of N_d functions in $\mathcal{F}$ such that $D(\cdot) = [d_1(\cdot), \dots, d_{N_d}(\cdot)]$ where $\{d_i\}_{i=1}^{N_d} \subset \mathcal{F}$, leading to the

[1]Some problem elements are similar to the elements used in the previous chapters. Here, we have repeated those notions for the reader's convenience.

EDMD optimization problem (cf. Section 2.4)

$$\underset{K}{\text{minimize}} \|D(Y) - D(X)K\|_F \tag{6.3}$$

which has the closed-form solution

$$K_{\text{EDMD}} = \text{EDMD}(D, X, Y) := D(X)^\dagger D(Y). \tag{6.4}$$

Consequently, given a function $f \in \text{span}(D)$ in the form of $f(\cdot) = D(\cdot)v_f$ for $v_f \in \mathbb{C}^{N_d}$, one can define the predictor function for $\mathcal{K}f$ as

$$\mathfrak{P}_{\mathcal{K}f}(\cdot) = D(\cdot)K_{\text{EDMD}}v_f. \tag{6.5}$$

Throughout this chapter, we rely on the following Assumption.

Assumption 6.2.1. ***(Full Rank Dictionary Matrices).*** *$D(X)$ and $D(Y)$ have full column rank.* □

Assumption 6.2.1 implies that the functions in D are linearly independent, i.e., they form a basis for span(D) and the data are diverse enough to distinguish between the elements of D. Moreover, Assumption 6.2.1 ensures that the optimization problem in (6.3) has a unique solution. In this chapter, we aim to address the following problem.

Problem 6.2.2. ***(Characterization of EDMD's Prediction Accuracy and the Dictionary's Quality).*** *Given a dictionary D and data matrices X and Y, under Assump-*

tion 6.2.1, we aim to provide a data-driven measure of the EDMD's accuracy and the dictionary's quality that

(i) only depends on $\mathrm{span}(D)$*,* X*, and* Y*, and is therefore invariant under the choice of basis for* $\mathrm{span}(D)$*, i.e., given* D' *as an alternative basis for* $\mathrm{span}(D)$*, the accuracy measures calculated based on* D *and* D' *are equal;*

(ii) provides a data-driven bound on the distance between $\mathcal{K}f$ *and its EDMD prediction* $\mathfrak{P}_{\mathcal{K}f}$ *for all functions* $f \in \mathrm{span}(D)$*;*

(iii) can be computed using a closed-form formula (for implementation in optimization solvers). □

6.3 Forward-Backward Temporal Consistency

Here, we take the first step towards finding an appropriate measure for EDMD's prediction accuracy by comparing the solutions of EDMD forward and backward in time[2]. Throughout this chapter, we use the following notation for forward and backward EDMD matrices

$$\begin{aligned} K_F &= \mathrm{EDMD}(D, X, Y) = D(X)^{\dagger} D(Y), \\ K_B &= \mathrm{EDMD}(D, Y, X) = D(Y)^{\dagger} D(X). \end{aligned} \tag{6.6}$$

[2]The idea of looking forward and backward in time has been considered in the literature for different purposes, such as improving DMD to deal with noisy data [DHWR16, AYB19] and identifying exact Koopman eigenfunctions in Chapter 3 and [HC22a] but, to the best of our knowledge, not for formally characterizing EDMD's prediction accuracy.

We particularly rely on the observation that if the dictionary spans a Koopman-invariant subspace, then $K_F K_B = I$. Otherwise, the forward and backward EDMD matrices will not be the inverse of each other. This motivates the definition of the consistency matrix.

Definition 6.3.1. ***(Consistency Matrix and Index).*** *Given dictionary D and data matrices X and Y, the* consistency matrix *is $M_C(D, X, Y) = I - K_F K_B$ and the* consistency index *is $\mathcal{I}_C(D, X, Y) = \operatorname{sprad}\big(M_C(D, X, Y)\big)$.* □

For convenience, we refer to $M_C(D, X, Y)$ and $\mathcal{I}_C(D, X, Y)$ as M_C and $\mathcal{I}_C$ when the context is clear. Next, we show that the eigenvalues of the consistency matrix are invariant under linear transformations of the dictionary.

Proposition 6.3.2. ***(Consistency Matrix's Spectrum is Invariant under Linear Transformation of Dictionary).*** *Let D and $\tilde{D}$ be two dictionaries such that $\tilde{D}(\cdot) = D(\cdot)R$, where R is an invertible matrix. Moreover, given data matrices X and Y let Assumption 6.2.1 hold. Then,*

(i) $M_C(\tilde{D}, X, Y) = R^{-1} M_C(D, X, Y) R$;

(ii) $\operatorname{spec}\big(M_C(D, X, Y)\big) = \operatorname{spec}\big(M_C(\tilde{D}, X, Y)\big)$.

Proof. Note that part (ii) directly follows from part (i) and the fact that similarity transformations preserve the eigenvalues. To show part (i), define for convenience,

$$K_F = D(X)^\dagger D(Y), \qquad K_B = D(Y)^\dagger D(X),$$
$$\tilde{K}_F = \tilde{D}(X)^\dagger \tilde{D}(Y), \qquad \tilde{K}_B = \tilde{D}(Y)^\dagger \tilde{D}(X).$$

We start by showing that $K_F K_B$ and $\tilde{K}_F \tilde{K}_B$ are similar. By definition, one can write

$$\tilde{K}_F \tilde{K}_B = \tilde{D}(X)^\dagger \tilde{D}(Y) \tilde{D}(Y)^\dagger \tilde{D}(X). \tag{6.7}$$

Moreover, given Assumption 6.2.1 and the definition of $\tilde{D}$, one can write

$$\tilde{D}(X)^\dagger = \big(\tilde{D}(X)^T \tilde{D}(X)\big)^{-1} \tilde{D}(X)^T = \big(R^T D(X)^T D(X) R\big)^{-1} R^T D(X)^T = R^{-1} D(X)^\dagger. \tag{6.8}$$

Equations (6.7)-(6.8), in conjunction with the fact that $\tilde{D}(Y)^\dagger \tilde{D}(Y) = D(Y)^\dagger D(Y)$ (cf. Lemma 6.5.1 in the appendix), imply $\tilde{K}_F \tilde{K}_B = R^{-1} K_F K_B R$ directly leading to the required identity following Definition 6.3.1. □

According to Proposition 6.3.2, the spectrum of the consistency matrix is a property of the data and the *vector space* spanned by the dictionary, as opposed to the dictionary itself. This property is consistent with the requirement in Problem 6.2.2(i). Next, we further investigate the eigendecomposition of the consistency matrix to extract useful information for our analysis.

Theorem 6.3.3. ***(Consistency Matrix's Properties).*** *Given Assumption 6.2.1, the consistency matrix $M_C(D, X, Y)$ has the following properties:*

(i) it is similar to a symmetric matrix;

(ii) it is diagonalizable with a complete set of eigenvectors;

(iii) $\operatorname{spec}\big(M_C(D, X, Y)\big) \subset [0, 1]$.

Proof. (i) Given Assumption 6.2.1, there exists an invertible matrix R such that the columns of $D(X)R$ are orthonormal. Define the dictionary $\tilde{D}(\cdot) = D(\cdot)R$. Note that $\tilde{D}(X)^T\tilde{D}(X) = I_{N_d}$ and hence $\tilde{D}(X)^\dagger = \big(\tilde{D}(X)^T\tilde{D}(X)\big)^{-1}\tilde{D}(X)^T = \tilde{D}(X)^T$. Using now the definition of the consistency matrix, we have

$$M_C(\tilde{D}, X, Y) = I - \tilde{D}(X)^T\tilde{D}(Y)\tilde{D}(Y)^\dagger\tilde{D}(X).$$

Noting that $\tilde{D}(Y)\tilde{D}(Y)^\dagger$ is symmetric, we deduce that $M_C(\tilde{D}, X, Y)$ is symmetric. Then (i) directly follows by the definition of R and Proposition 6.3.2(i).

(ii) The proof directly follows from part (i) and the fact that symmetric matrices are diagonalizable and have a complete set of eigenvectors.

(iii) From part (i), we deduce that $M_C(D, X, Y)$ has real eigenvalues. Since $M_C(D, X, Y) = I - D(X)^\dagger D(Y)D(Y)^\dagger D(X)$, we only need to show

$$\operatorname{spec}\big(D(X)^\dagger D(Y)D(Y)^\dagger D(X)\big) \subset [0, 1]. \tag{6.9}$$

Consider an eigenvector $v \in \mathbb{R}^{N_d} \setminus \{0\}$ with eigenvalue μ, i.e., $D(X)^\dagger D(Y)D(Y)^\dagger D(X)v = \mu v$. Multiplying both sides from the left by $D(X)$ and defining $w = D(X)v$ leads to $D(X)D(X)^\dagger D(Y)D(Y)^\dagger w = \mu w$. Next, by multiplying this equation from the left by w^T we have

$$w^T D(X)D(X)^\dagger D(Y)D(Y)^\dagger w = \mu\|w\|_2^2. \tag{6.10}$$

Now, the fact that $D(X)D(X)^\dagger$ is symmetric and represents the orthogonal projection operator on $\mathcal{R}(D(X))$, in conjunction with $w \in \mathcal{R}(D(X))$, allows us to write $w^T D(X)D(X)^\dagger = w^T$. This identity combined with (6.10) leads to

$$\mu = \frac{w^T D(Y)D(Y)^\dagger w}{\|w\|_2^2}.$$

Hence, $\lambda_{\min}(D(Y)D(Y)^\dagger) \leq \mu \leq \lambda_{\max}(D(Y)D(Y)^\dagger)$. However, since $D(Y)D(Y)^\dagger$ is an orthogonal projection operator, we have $\operatorname{spec}(D(Y)D(Y)^\dagger) \subset [0,1]$. Consequently, $\mu \in [0,1]$, leading to (6.9) and concluding the proof. □

As a consequence of Theorem 6.3.3, the consistency matrix is similar to a positive semidefinite matrix. The larger the eigenvalues of M_C, the more inconsistent the forward and backward EDMD models get. Also, from Theorem 6.3.3, $\mathcal{I}_C = \lambda_{\max}(M_C) \in [0,1]$. Intuitively, the consistency index determines the quality of the subspace spanned by the dictionary and the prediction accuracy of EDMD on it. This is what we formalize in the next section.

6.4 Consistency Index Determines EDMD's Prediction Accuracy on Data

Our main result states that the square root of the consistency index is a tight upper bound for the relative root mean square prediction error of EDMD.

Theorem 6.4.1. ***($\sqrt{\mathcal{I}_C}$ Bounds the Relative Root Mean Square Error (RRMSE)***

of EDMD). *For dictionary D and data matrices X, Y, under Assumption 6.2.1,*

$$\begin{aligned}\text{RRMSE}_{\max} &:= \max_{f \in \operatorname{span}(D)} \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} |\mathcal{K}f(x_i) - \mathfrak{P}_{\mathcal{K}f}(x_i)|^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} |\mathcal{K}f(x_i)|^2}} \\ &= \sqrt{\mathcal{I}_C(D, X, Y)}.\end{aligned}$$

where the predictor $\mathfrak{P}_{\mathcal{K}f}$ is defined in (6.5). □

Note that the combination of Definition 6.3.1 and Theorems 6.3.3 and 6.4.1 mean that $\sqrt{\mathcal{I}_C(D, X, Y)}$ satisfies all the requirements in Problem 6.2.2. Before proving the result, we first remark its importance regarding function predictions.

Remark 6.4.2. ***($\sqrt{\mathcal{I}_C}$ Determines the Relative L_2-norm Error of EDMD's Prediction under Empirical Measure).*** Given that the elements of span(D) and their composition with T are measurable and considering the empirical measure $\mu_X = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}$, where δ_{x_i} is the Dirac measure defined based on the ith row of X, one can rewrite $\text{RRMSE}_{\max}$ as

$$\text{RRMSE}_{\max} = \max_{f \in \operatorname{span}(D)} \frac{\|\mathcal{K}f - \mathfrak{P}_{\mathcal{K}f}\|_{L_2(\mu_X)}}{\|\mathcal{K}f\|_{L_2(\mu_X)}} = \sqrt{\mathcal{I}_C}. \qquad \square$$

To prove Theorem 6.4.1, we first provide the following alternative expression of the consistency index.

Theorem 6.4.3. ***(Consistency Index and Difference of Projections).*** *Given As-*

sumption 6.2.1,

$$\sqrt{\mathcal{I}_C(D,X,Y)} = \text{sprad}\left(D(Y)D(Y)^\dagger - D(X)D(X)^\dagger\right).$$

Proof. From Theorem 6.3.3, we have $\mathcal{I}_C = \lambda_{\max}(M_C)$. We use the following notation throughout the proof,

$$\lambda_{\max} = \mathcal{I}_C,\ \mathcal{P}_{D(X)} = D(X)D(X)^\dagger,\ \mathcal{P}_{D(Y)} = D(Y)D(Y)^\dagger.$$

Note that $\mathcal{P}_{D(X)}$ and $\mathcal{P}_{D(Y)}$ are projection operators on $\mathcal{R}(D(X))$ and $\mathcal{R}(D(Y))$, respectively. By Definition 6.3.1, given an eigenvalue $\lambda \in [0,1]$ of M_C with eigenvector $v \neq 0$,

$$M_C v = \lambda v \Leftrightarrow K_F K_B v = (1-\lambda)v. \tag{6.11}$$

We consider the cases (i) $\lambda_{\max} = 0$, (ii) $\lambda_{\max} = 1$, and (iii) $\lambda_{\max} \in (0,1)$ separately.

Case (i): $\lambda_{\max} = 0$. In this case, from Theorem 6.3.3, we deduce $M_C = 0$. Consequently, $K_F K_B = I$. By multiplying both sides from the left by $D(X)$ and collecting the terms, we have $\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}D(X) = D(X)$. Hence, one can write

$$\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}z = z,\ \forall z \in \mathcal{R}(D(X)). \tag{6.12}$$

Moreover, based on the analysis in [AHT85], one can deduce that

$$\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}w = w \Leftrightarrow w \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y)). \tag{6.13}$$

Using (6.12)-(6.13), one can write $\mathcal{R}(D(X)) \subseteq \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))$ and consequently $\mathcal{R}(D(X)) \subseteq \mathcal{R}(D(Y))$. By a similar argument as above and swapping K_F with K_B and $D(X)$ with $D(Y)$, one can also deduce $\mathcal{R}(D(Y)) \subseteq \mathcal{R}(D(X))$. Hence, $\mathcal{R}(D(X)) = \mathcal{R}(D(Y))$. Moreover, since the orthogonal projection on a subspace is unique, we have $\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)} = 0$, concluding the proof for this part.

Case (ii): $\lambda_{\max} = 1$. By setting $\lambda = \lambda_{\max}$ in (6.11), multiplying both sides from the left by $D(X)$, defining $w := D(X)v$, we have

$$\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}w = 0. \tag{6.14}$$

Hence, noting that $w \neq 0$ (based on Assumption 6.2.1 and the fact that $v \neq 0$), we can deduce it is an eigenvector of $\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}$ with eigenvalue 0. We show next that $w \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))^{\perp}$. One can uniquely decompose $w \in \mathcal{R}(D(X))$ as $w = w_{D(Y)} + w_{D(Y)^{\perp}}$, where $w_{D(Y)} \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))$ and $w_{D(Y)^{\perp}} \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))^{\perp}$. Noting that $\mathcal{P}_{D(Y)}w_{D(Y)^{\perp}} = 0$ and $\mathcal{P}_{D(Y)}w_{D(Y)} = w_{D(Y)}$, we get from (6.14) that $\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}w = \mathcal{P}_{D(X)}w_{D(Y)} = 0$. Since $w_{D(Y)} \in \mathcal{R}(D(X))$, we deduce $w_{D(Y)} = 0$ and consequently,

$$w = w_{D(Y)^{\perp}} \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))^{\perp}.$$

Therefore, $(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)})w = -w$ and, given that $w \neq 0$, we deduce that $\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}$ has an eigenvalue equal to -1. Since $\mathrm{spec}(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}) \subset [-1, 1]$, cf. [AHT85, Lemma 1], we conclude $\mathrm{sprad}(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}) = 1$. The proof concludes by noting that $\mathcal{I}_C = \lambda_{\max} = 1$.

Case (iii): $\lambda_{\max} \in (0, 1)$**.** Using Lemma 6.5.2 and the closed-form expressions of K_F, K_B, $\mathcal{P}_{D(X)}$, and $\mathcal{P}_{D(Y)}$,

$$\mathrm{spec}_{\neq 0}(K_F K_B) = \mathrm{spec}_{\neq 0}(\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)}). \tag{6.15}$$

Given $\mu \in (0, 1)$, from [AHT85, Theorems 1-2], we have $\mu \in \mathrm{spec}_{\neq 0}(\mathcal{P}_{D(X)}\mathcal{P}_{D(Y)})$ if and only if $\{\pm\sqrt{1-\mu}\} \subset \mathrm{spec}_{\neq 0}\left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right)$. By setting $\mu = 1 - \lambda$, $\lambda \in (0, 1)$, one can use this identity in conjunction with (6.11) and (6.15) to write

$$\lambda \in \mathrm{spec}_{\neq 0}(M_C) \Leftrightarrow \{\pm\sqrt{\lambda}\} \subset \mathrm{spec}_{\neq 0}\left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right).$$

This, in conjunction with [AHT85, Theorem 1] and the fact that $\mathrm{sprad}\left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right) \leq 1$ (cf. [AHT85, Lemma 1]), shows that if $\mathrm{sprad}\left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right) < 1$, then the result holds. To conclude the proof, we just need to show that $\mathrm{sprad}\left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right) = 1$ cannot hold, which we show by contradiction. Suppose this is the case, then at least one of the following holds:

(i) $\exists w_1 \in \mathbb{R}^N \setminus \{0\};\ \left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right)w_1 = -w_1$,

(ii) $\exists w_2 \in \mathbb{R}^N \setminus \{0\};\ \left(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\right)w_2 = w_2$.

For case (i), note that based on the analysis in [AHT85], we have

$$w_1 \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))^\perp.$$

Now, consider the vector $p_1 \neq 0$ with $w_1 = D(X)p_1$. Consequently, one can write $K_F K_B p_1 = D(X)^\dagger \mathcal{P}_{D(Y)} w_1 = 0$, where in the last equality we have used $w_1 \perp \mathcal{R}(D(Y))$. However, this implies that $M_C p_1 = p_1$, contradicting the fact that $\lambda_{\max} \in (0,1)$.

For case (ii), note that $w_2 \in \mathcal{R}(D(Y)) \cap \mathcal{R}(D(X))^\perp$ (see e.g., [AHT85]). Consider the vector space $\mathcal{S} = \mathcal{R}(D(Y)) \cap \mathcal{R}(w_2)^\perp$. Clearly $\dim \mathcal{S} < \dim \mathcal{R}(D(Y)) = \dim \mathcal{R}(D(X))$. Consequently, there exists a non-zero vector $w^* \in \mathcal{R}(D(X))$ such that $w^* \perp \mathcal{S}$. Also, $w^* \perp \mathcal{R}(w_2)$ since $w_2 \perp \mathcal{R}(D(X))$. Hence, by noting that $\mathcal{R}(D(Y))$ is the direct sum of $\mathcal{S}$ and $\mathcal{R}(w_2)$, one can conclude $w^* \in \mathcal{R}(D(X)) \cap \mathcal{R}(D(Y))^\perp$ and, as a result, we have $\big(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}\big)w^* = -w^*$. Since w^* satisfies the identity in case (i), the proof follows by replacing w_1 with w^* in the proof of case (i). □

We are finally ready to prove Theorem 6.4.1.

Proof. ***(Theorem 6.4.1).*** We use the following notation throughout the proof: $\mathcal{P}_{D(X)} = D(X)D(X)^\dagger$, $\mathcal{P}_{D(Y)} = D(Y)D(Y)^\dagger$, and $s = \text{sprad}(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)})$. Note that, from Theorem 6.4.3, $s = \sqrt{\mathcal{I}_C(D,X,Y)}$. Given an arbitrary function $f(\cdot) = D(\cdot)v_f \in \text{span}(D)$, with $v_f \in \mathbb{C}^{N_d}$, one can use (2.3), the predictor (6.5) with $K_F = K_{\text{EDMD}} = \text{EDMD}(D,X,Y)$, and the relationship between the rows of X and Y in (6.2) to write

$$\begin{aligned}\mathrm{RRMSE}_f &:= \frac{\sqrt{\sum_{i=1}^{N} |D(y_i)v_f - D(x_i)K_F v_f|^2}}{\sqrt{\sum_{i=1}^{N} |D(y_i)v_f|^2}} \\ &= \frac{\|D(Y)v_f - D(X)K_F v_f\|_2}{\|D(Y)v_f\|_2} \qquad (6.16)\end{aligned}$$

Noting that $D(Y) = \mathcal{P}_{D(Y)}D(Y)$, one can write

$$\begin{aligned}\|D(Y)v_f - D(X)K_F v_f\|_2 &= \|(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)})D(Y)v_f\|_2 \\ &\leq s\|D(Y)v_f\|_2, \qquad (6.17)\end{aligned}$$

where the last inequality holds since the matrix $\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}$ is symmetric and therefore its spectral radius is equal to its induced 2-norm. Based on (6.16)-(6.17), we have $\mathrm{RRMSE}_f \leq s$. Hence, by definition of $\mathrm{RRMSE}_{\max}$ in the statement of the result, we have

$$\mathrm{RRMSE}_{\max} = \max_{f \in \mathrm{span}(D)} \mathrm{RRMSE}_f \leq s \qquad (6.18)$$

Now, we prove that the equality in (6.18) holds. We consider three cases: (i) $s = 0$ (ii) $s = 1$ or (iii) $s \in (0, 1)$.

Case (i): Since $\mathrm{RRMSE}_{\max} \geq 0$ by definition, in this case $\mathrm{RRMSE}_{\max} = 0$ follows directly.

Case (ii): In this case, there exists a vector[3] $p^* \in \mathcal{R}(D(Y)) \cap \mathcal{R}(D(X))^{\perp}$. Let v^* be

[3]The argument for the existence of p^* is similar (by swapping $D(X)$ and $D(Y)$) to the argument used for the existence of vectors w_1 and w^* in the proof of Theorem 6.4.3 (Case (iii)). We omit this argument for space reasons.

such that $p^* = D(Y)v^*$. Using (6.17) for v^* instead of v_f, and the properties of p^*, one can write $\|D(Y)v^* - D(X)K_F v^*\|_2 = \|\mathcal{P}_{D(Y)} D(Y)v^*\|_2 = \|D(Y)v^*\|_2$. Hence, for the function $f^*(\cdot) = D(\cdot)v^* \in \operatorname{span}(D)$, one can use (6.16) to see that $\text{RRMSE}_{f^*} = 1 = s$. Hence, equality holds in (6.18).

Case (iii): In this case $s \in (0, 1)$ and based on [AHT85, Theorem 1], the matrix $\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)}$ has two eigenvalues $\pm s$ with corresponding orthogonal eigenvectors $v_{+s}, v_{-s} \in \mathbb{R}^{N_d}$. Moreover, based on [AHT85, Theorem 1(a)], $\mathcal{P}_{D(Y)} v_{+s} \in \operatorname{span}\{v_{+s}, v_{-s}\}$. Hence, for some $\alpha, \beta \in \mathbb{R}$, we have

$$q^* := \mathcal{P}_{D(Y)} v_{+s} = \alpha v_{+s} + \beta v_{-s}.$$

Let r^* be such that $q^* = D(Y)r^*$. Now, based on the first part of (6.17) for r^* instead of v_f, we have

$$\begin{aligned}
&\|D(Y)r^* - D(X)K_F r^*\|_2 = \|(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)})D(Y)r^*\|_2 \\
&= \|(\mathcal{P}_{D(Y)} - \mathcal{P}_{D(X)})(\alpha v_{+s} + \beta v_{-s})\|_2 = s\|\alpha v_{+s} - \beta v_{-s}\|_2 \\
&= s\|\alpha v_{+s} + \beta v_{-s}\|_2 = s\|D(Y)r^*\|_2, \qquad (6.19)
\end{aligned}$$

where in the third and fourth equalities we have used the definition of v_{+s} and v_{-s} and their orthogonality. Now, for the function $g^*(\cdot) = D(\cdot)r^* \in \operatorname{span}(D)$, one can use (6.16) to see that $\text{RRMSE}_{g^*} = s$. Hence, the equality in (6.18) holds, and this concludes the proof. □

Remark 6.4.4. ***(Working with Consistency Matrix is More Efficient than the***

Difference of Projections). According to Theorems 6.4.1 and 6.4.3, one can use the consistency matrix $M_C \in \mathbb{R}^{N_d \times N_d}$ or the difference of projections $D(Y)D(Y)^\dagger - D(X)D(X)^\dagger \in \mathbb{R}^{N \times N}$ interchangeably to compute the relative root mean square error. However, note that the size of the consistency matrix depends on the dictionary N_d, while the size of the difference of projections depends on the size of data N. In most practical settings $N \gg N_d$, and consequently, working with the consistency matrix is more efficient. In fact, given moderate to large data sets, even saving the difference of projections matrix in the memory may be infeasible. The calculation of the consistency matrix requires solving two least-squares problems, which can be done recursively for large data sets. $\square$

6.5 Chapter Appendix

Here, we recall two results that are used in the proofs.

Lemma 6.5.1. *Let $B_1, B_2 \in \mathbb{R}^{m \times n}$ be matrices such that $\mathcal{R}(B_1) = \mathcal{R}(B_2)$. Then $B_1 B_1^\dagger = B_2 B_2^\dagger$.* $\square$

The proof of follows from the uniqueness of the orthogonal projection operator on a subspace.

Lemma 6.5.2. ***([Ber09, Proposition 4.4.10]).*** *Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times m}$. Then, $\operatorname{spec}_{\neq 0}(AB) = \operatorname{spec}_{\neq 0}(BA)$.* $\square$

Acknowledgements

This chapter is taken, in part, from the work [HC22b] which has been submitted for publication as ‘Temporal Forward-Backward Consistency, Not Residual Error, Measures the Prediction Accuracy of Extended Dynamic Mode Decomposition’ by M. Haseli and J. Cortés, in IEEE Control Systems Letters. The dissertation author was the primary investigator and author of this paper. The work in this chapter was partially supported by ONR Award N00014-18-1-2828.

Chapter 7

Conclusions

The central theme of this dissertation is using the Koopman operators associated with nonlinear systems to extract meaningful information from data. We have provided several algebraic methods to efficiently identify Koopman eigenfunctions and invariant subspaces containing important information about the system's behavior. Prompted by practical challenges such as the need for fast and real-time computations as well as memory limitations, we have provided methods suitable for parallel processing and algorithms that can handle large and streaming data sets. For many nonlinear systems, capturing complete information on finite-dimensional Koopman-invariant subspaces might not be possible. To address this issue, we have provided methods to tune the balance between the accuracy and expressiveness of models. Finally, even though our main focus has been on algebraic methods as opposed to optimization algorithms, we have provided a data-driven measure for the subspace quality, which can be used as an effective cost function in optimization-based algorithms.

7.1 Summary

Chapter 3 studies the characterization of Koopman-invariant subspaces and Koopman eigenfunctions associated to a dynamical system by means of data-driven methods. We have shown that the application of Extended Dynamic Mode Decomposition algorithm over a given dictionary forward and backward in time fully characterizes whether a function evolves linearly in time according to the available data. Building on this result, and under dense sampling, we have established that functions satisfying this condition are Koopman eigenfunctions almost surely. We have developed the Symmetric Subspace Decomposition algorithm to identify the maximal Koopman-invariant subspace in the span of the given dictionary and formally characterized its correctness. Finally, we have developed extensions to scenarios with large and streaming data sets, where the algorithm refines its output as new data becomes available, and to scenarios where the original dictionary does not contain sufficient informative eigenfunctions, in which case the algorithm obtains approximations of the Koopman eigenfunctions and invariant subspaces.

In Chapter 4, we have proposed a parallel counterpart for the Symmetric Subspace Decomposition algorithm presented in Chapter 3. Our strategy termed Parallel Symmetric Subspace Decomposition (P-SSD) runs in parallel over a network of processors that communicate over a digraph. We have characterized the algorithm's convergence and complexity properties. We have identified conditions on the network connectivity that ensure that the algorithm's output coincides with that of Symmetric Subspace Decomposition when run over the whole set of data snapshots. The parallel nature of P-SSD makes it run significantly faster than its centralized counterpart. We have also established the robustness of P-SSD against communication failures and packet drops.

In Chapter 5, we have presented the Tunable Symmetric Subspace Decomposition (T-SSD) algorithm, a data-driven strategy that employs data snapshots from an unknown dynamical system to refine a given dictionary of functions, yielding a subspace close to being invariant under the Koopman operator. A design parameter allows to balance the prediction accuracy and expressiveness of the algorithms' output, which always contains the maximal Koopman-invariant subspace and all Koopman eigenfunctions in the span of the original dictionary. The proposed algorithm generalizes both the Extended Dynamic Mode Decomposition method as well as the Symmetric Subspace Decomposition algorithm presented in Chapter 3.

Chapter 6 considers the problem of assessing a subspace's quality for Koopman approximations. We have investigated the residual error of Extended Dynamic Mode Decomposition which is generally used for subspace assessment and reached the conclusion that it is not an effective tool due to its sensitivity to the choice of the basis for the subspace. As a remedy, we have introduced the concept of consistency index, a data-driven measure that quantifies the accuracy of the Extended Dynamic Mode Decomposition method on a finite-dimensional functional space generated by a dictionary of functions. The consistency index is calculated based on comparing the solutions of Extended Dynamic Mode Decomposition ran forward and backward in time. The consistency index is invariant under the choice of basis of the functional space, is computable in closed form, and provides a bound for the relative root mean squared error of all functions predictions in the subspace.

7.2 Future Work

This dissertation has laid the groundwork for many research possibilities which we will pursue in the future. Here, we discuss our short-term goals and long-term vision.

The algebraic algorithms proposed in Chapters 3-5 open up exciting research avenues to explore. Our future research will investigate the integration of our algorithms with machine learning techniques to streamline the subspace identification process while providing accuracy and convergence guarantees for the outputs. Moreover, we aim to extend our methods to use known information about the system which can speed up the subspace learning process and lead to more accurate models. In addition, we aim to explore the connection of our data-driven methods to model-based approximation and reduction techniques.

Another interesting research direction focuses on distributed algorithms to learn Koopman eigenfunctions and invariant subspaces. Such schemes can be fully distributed and asynchronous. We have already explored parallel algorithms in Chapter 4. Our future work will employ other distributed strategies for the agents to sample data independently and search through different subspaces. Moreover, other techniques such as event-triggered methods can be employed to reduce the computational cost.

Exploring potential applications is an important future direction for our research. Our work has been focused on identifying Koopman eigenfunctions and invariant subspaces. However, one can make several connections between Koopman eigenfunctions and practical control problems. The literature has explored the connection between Koopman eigenfunctions and the stability properties of attractors. In particular, there are instances where Koopman eigenfunctions lead to construction of Lyapunov functions. We aim to build on such results in the literature and design data-driven Koopman-based methods to construct Lyapunov functions and provide stability guarantees. Another interesting direction would be to use our work to design controllers. There are several Koopman-based methods in the literature for control applications. The general theme among them is the lack of performance guarantees which is a direct result of the lack of guarantees in Koopman operator modeling. Given that our algorithms provide accuracy and convergence guarantees, it is natural to think about using them to perform control tasks in a principled manner accompanied by convergence and stability certification.

Bibliography

[AdlTM17] I. Abraham, G. de la Torre, and T. Murphey. Model-based control using Koopman operators. In *Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[AHT85] W. N. Anderson Jr, E. J. Harner, and G. E. Trapp. Eigenvalues of the difference and product of projections. *Linear and Multilinear Algebra*, 17(3-4):295–299, 1985.

[AK17] A. Alla and J. N. Kutz. Nonlinear model order reduction via dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 39(5):B778–B796, 2017.

[AM19] S. Anantharamu and K. Mahesh. A parallel and streaming dynamic mode decomposition algorithm with finite precision error analysis for large data. *Journal of Computational Physics*, 380:355–377, 2019.

[AMS09] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds.* Princeton University Press, 2009.

[AYB19] O. Azencot, W. Yin, and A. Bertozzi. Consistent dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(3):1565–1585, 2019.

[BBK$^+$21] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche. Koopmanizingflows: Diffeomorphically learning stable Koopman operators. *arXiv preprint arXiv:2112.04085*, 2021.

[BBPK16] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS One*, 11(2):1–19, 2016.

[BCM09] F. Bullo, J. Cortés, and S. Martinez. *Distributed Control of Robotic Networks.* Applied Mathematics Series. Princeton University Press, 2009.

[Ber09] D. S. Bernstein. *Matrix Mathematics.* Princeton University Press, 2 edition, 2009.

[BMM12] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanism. *Chaos*, 22(4):047510, 2012.

[BPK16] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[BRV19] D. Bruder, C. D. Remy, and R. Vasudevan. Nonlinear system identification of soft robot dynamics using Koopman operator theory. In *IEEE Int. Conf. on Robotics and Automation*, pages 6244–6250, Montreal, Canada, May 2019.

[Cor08] J. Cortés. Distributed algorithms for reaching consensus on general functions. *Automatica*, 44(3):726–737, 2008.

[CTR12] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of dynamic mode decomposition: boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012.

[CV17] S. Le Clainche and J. M. Vega. Higher-order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017.

[DHWR16] S. T. M. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):42, 2016.

[EY36] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[FCAB20] C. Folkestad, Y. Chen, A. D. Ames, and J. W. Burdick. Data-driven safety-critical control: Synthesizing control barrier functions with Koopman operators. *IEEE Control Systems Letters*, 5(6):2012–2017, 2020.

[Fol99] G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley, New York, 2nd edition, 1999.

[FYR$^+$21] F. Fan, B. Yi, D. Rye, G. Shi, and I.R. Manchester. Learning stable Koopman embeddings. *arXiv preprint arXiv:2110.06509*, 2021.

[GP21] D. Goswami and D. A. Paley. Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A Koopman spectral approach. *IEEE Transactions on Automatic Control*, 2021. To appear.

[HC19a] M. Haseli and J. Cortés. Approximating the Koopman operator using noisy data: noise-resilient extended dynamic mode decomposition. In *American Control Conference*, pages 5499–5504, Philadelphia, PA, July 2019.

[HC19b] M. Haseli and J. Cortés. Efficient identification of linear evolutions in nonlinear vector fields: Koopman invariant subspaces. In *IEEE Conf. on Decision and Control*, pages 1746–1751, Nice, France, December 2019.

[HC21a] M. Haseli and J. Cortés. Generalizing dynamic mode decomposition: balancing accuracy and expressiveness in Koopman approximations. *Automatica*, 2021. Submitted.

[HC21b] M. Haseli and J. Cortés. Parallel learning of Koopman eigenfunctions and invariant subspaces for accurate long-term prediction. *IEEE Transactions on Control of Network Systems*, 8(4):1833–1845, 2021.

[HC22a] M. Haseli and J. Cortés. Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric Subspace Decomposition. *IEEE Transactions on Automatic Control*, 67(7):3442–3457, 2022.

[HC22b] M. Haseli and J. Cortés. Temporal forward-backward consistency, not residual error, measures the prediction accuracy of extended dynamic mode decomposition. *https://arxiv.org/abs/2207.07719*, 2022.

[HMV18] B. Huang, X. Ma, and U. Vaidya. Feedback stabilization using Koopman operator. In *IEEE Conf. on Decision and Control*, pages 6434–6439, Miami Beach, FL, December 2018.

[HRDC17] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics*, 31(4):349–368, 2017.

[HWR14] M. S. Hemati, M. O. Williams, and C. W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 2014.

[JSN14] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.

[JT19] R. M. Jungers and P. Tabuada. Non-local linearization of nonlinear differential equations via polyflows. In *American Control Conference*, pages 1906–1911, Philadelphia, PA, 2019.

[JY18] C. A. Johnson and E. Yeung. A class of logistic functions for approximating state-inclusive Koopman operators. In *American Control Conference*, pages 4803–4810, Milwaukee, WI, June 2018. IEEE.

[KGB$^+$17] B. Kramer, P. Grover, P. Boufounos, S. Nabi, and M. Benosman. Sparse sensing and DMD-based identification of flow regimes and bifurcations in complex flows. *SIAM Journal on Applied Dynamical Systems*, 16(2):1164–1196, 2017.

[KKB21] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2(3):035023, 2021.

[KM18a] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[KM18b] M. Korda and I. Mezić. On convergence of extended dynamic mode decomposition to the Koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

[KM20] M. Korda and I. Mezic. Optimal construction of Koopman eigenfunctions for prediction and control. *IEEE Transactions on Automatic Control*, 65(12):5114–5129, 2020.

[KN32] B. O. Koopman and J. V. Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, 1932.

[KNK$^+$18] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé. Data-driven model reduction and transfer operator approximation. *Journal of Nonlinear Science*, 28(3):985–1010, 2018.

[KNP$^+$20] S. Klus, F. Nüske, S. Peitz, J. H. Niemann, C. Clementi, and C. Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[Koo31] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[LDBK17] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos*, 27(10):103111, 2017.

[LKB18] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):1–10, 2018.

[LM13] Y. Lan and I. Mezić. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013.

[LMNB17] J. Liu, A. S. Morse, A. Nedić, and T. Başar. Exponential convergence of a distributed algorithm for solving linear algebraic equations. *Automatica*, 83:37–46, 2017.

[LT20] H. Lu and D. M. Tartakovsky. Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 42(3):A1639–A1662, 2020.

[LWC19] X. Li, S. Wang, and Y. Cai. Tutorial: Complexity analysis of singular value decomposition and its variants. *arXiv preprint arXiv:1906.12085*, 2019.

[LWJ21] Y. Lian, R. Wang, and C. N. Jones. Koopman based data-driven predictive control. *arXiv preprint arXiv:2102.05122*, 2021.

[Lyn97] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1997.

[MCTM21] G. Mamakoukas, M. L. Castano, X. Tan, and T. D. Murphey. Derivative-based Koopman operators for real-time control of robotic systems. *IEEE Transactions on Robotics*, 2021.

[Mez05] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.

[MG16] A. Mauroy and J. Goncalves. Linear identification of nonlinear systems: A lifting technique based on the Koopman operator. In *IEEE Conf. on Decision and Control*, pages 6500–6505, Las Vegas, NV, December 2016.

[MG19] A. Mauroy and J. Goncalves. Koopman-based lifting techniques for nonlinear systems identification. *IEEE Transactions on Automatic Control*, 2019. To appear.

[MH07] I. Markovsky and S. Van Huffel. Overview of total least-squares methods. *Signal processing*, 87(10):2283–2302, 2007.

[Mir60] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

[MM12] J. E. Marsden and M. McCracken. *The Hopf bifurcation and its applications*, volume 19. Springer Science & Business Media, 2012.

[MM16] A. Mauroy and I. Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, 2016.

[MMM13] A. Mauroy, I. Mezić, and J. Moehlis. Isostables, isochrons, and Koopman spectrum for the action-angle representation of stable fixed point dynamics. *Physica D: Nonlinear Phenomena*, 261:19–30, 2013.

[NM18] M. Netto and L. Mili. A robust data-driven Koopman Kalman filter for power systems dynamic state estimation. *IEEE Transactions on Power Systems*, 33(6):7228–7237, 2018.

[NPP$^+$21] F. Nüske, S. Peitz, F. Philipp, M. Schaller, and K. Worthmann. Finite-data error bounds for Koopman-based prediction and control. *arXiv preprint arXiv:2108.07102*, 2021.

[Pel00] D. Peleg. *Distributed Computing. A Locality-Sensitive Approach.* Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[PK19] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. *Automatica*, 106:184–191, 2019.

[RMB$^+$09] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[Sch10] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[SE17] A. Sootla and D. Ernst. Pulse-based control using Koopman operator under parametric uncertainty. *IEEE Transactions on Automatic Control*, 63(3):791–796, 2017.

[SK21] L. Shi and K. Karydis. Enhancement for robustness of Koopman operator-based data-driven mobile robotic systems. *arXiv preprint arXiv:2103.00812*, 2021.

[SVR16] S. Sinha, U. Vaidya, and R. Rajaram. Operator theoretic framework for optimal placement of sensors and actuators for control of nonequilibrium dynamics. *Journal of Mathematical Analysis and Applications*, 440(2):750–772, 2016.

[SWMB17] A. Surana, M. O. Williams, M. Morari, and A. Banaszuk. Koopman operator framework for constrained state estimation. In *IEEE Conf. on Decision and Control*, pages 94–101, Melbourne, Australia, 2017.

[TB97] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[TKY17] N. Takeishi, Y. Kawahara, and T. Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Conference on Neural Information Processing Systems*, pages 1130–1140, 2017.

[TRL$^+$14] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[WKR15] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[YKH19] E. Yeung, S. Kundu, and N. Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *American Control Conference*, pages 4832–4839, Philadelphia, PA, July 2019.

[ZB22a] V. Zinage and E. Bakolas. Koopman operator based modeling for quadrotor control on SE(3). *IEEE Control Systems Letters*, 6:752–757, 2022.

[ZB22b] V. Zinage and E. Bakolas. Neural Koopman Lyapunov control. *arXiv preprint arXiv:2201.05098*, 2022.

[ZRDC19] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.

[ZZ21] C. Zhang and E. Zuazua. A quantitative analysis of Koopman operator methods for system identification and predictions. *https://hal.archives-ouvertes.fr/hal-03278445*, 2021.