Learning Koopman Eigenfunctions and Invariant Subspaces from Data: Symmetric Subspace Decomposition

Masih Haseli and Jorge Cortés

***Abstract*—This paper develops data-driven methods to identify eigenfunctions of the Koopman operator associated to a dynamical system and subspaces that are invariant under the operator. We build on Extended Dynamic Mode Decomposition (EDMD), a data-driven method that finds a finite-dimensional approximation of the Koopman operator on the span of a predefined dictionary of functions. We propose a necessary and sufficient condition to identify Koopman eigenfunctions based on the application of 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.**

I. Introduction

Driven by advances in processing, data storage, cloud services, and algorithms, the world has witnessed in recent years a revolution in data-driven learning, analysis, and control of dynamical phenomena. State-space, probabilistic, and neural network models are among the most popular methods to model dynamical systems. With sufficient a priori information about the dynamics, state-space methods can provide closed-form analytic models that describe accurately the dynamical behavior. Such models, however, are generally nonlinear and their analytical study becomes arduous for moderate to high-dimensional systems. Probabilistic approaches, on the other hand, provide an alternative description that is conductive to dealing with incomplete information about the underlying dynamics. However, under such approaches, deriving mathematical guarantees may be hard, if not impossible. Neural networks can describe the dynamics with high accuracy given enough data. The models acquired by neural networks are highly nonlinear and difficult to study analytically. Hence, even though they can be successful in predicting the behavior of the system, they often do not provide a deeper understanding into their dynamics. 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 paper develops data-driven methods to identify such subspaces.

Literature Review: The Koopman operator [2], [3] 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 functional space. Being a linear operator enables one to use its spectral properties to capture and predict the behavior of nonlinear dynamical systems [4]–[6]. This leads to a wide variety of applications including state estimation [7], [8], system identification [9]–[11], sensor and actuator placement [12], model reduction [13], [14], control [15]–[21], and robotics [22], [23]. Moreover, the eigenfunctions of the Koopman operator play an important role in stability analysis of nonlinear systems [24]. 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 methods and invariant-subspace 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 [25]. DMD uses linear algebraic methods to form a linear model from time series data. The work [26] explores the properties of DMD and its connection with the Koopman operator, and [27] generalizes it to work with non-sequential data snapshots. Several extensions perform online computations to work with streaming datasets [28]–[30], account for the effect of measurement noise on data [31], [32], promote sparsity [33], and consider time-lagged data snapshots [34]. Extended Dynamic Mode Decomposition (EDMD) [35] is an important variations of DMD that lifts the states of the system

This work was supported by ONR Award N00014-18-1-2828.

A preliminary version of this work appeared as [1] at the IEEE Conference on Decision and Control.

Masih Haseli and Jorge Cortés are with Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {mhaseli,cortes}@ucsd.edu

to a (generally higher-dimensional) functional space using a predefined dictionary of functions and finds the projection of the Koopman operator on that subspace. The work [36] studies the convergence properties of EDMD to the Koopman operator as the number of data snapshots and dictionary elements go to infinity. EDMD is specifically designed to work with exact data and experiments and simulations show that it may not work well with noisy data. Our previous work [37] presented a noise-resilient extension of EDMD able to work with data corrupted with measurement noise. The basic lifting idea of EDMD can also be combined with known information about the dynamics to increase the accuracy of the model [38]. The aforementioned 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. The works [39]–[42] provide approaches to find functions that span Koopman-invariant subspaces using neural networks. Moreover, since Koopman eigenfunctions span Koopman-invariant subspaces, one can use the empirical methods provided in [19], [43] to approximate the Koopman eigenfunctions and consequently the invariant subspaces. Moreover, the work in [44] provides theoretical and empirical results based on multi-step predictions to find Koopman eigenfunctions. Interestingly, note that none of the aforementioned methods provide mathematical guarantees for the identified functions to be Koopman eigenfunctions.

Statement of Contributions: 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 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 in 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 issue, we provide an extension of SSD, termed Approximated-SSD, enabling us to approximate Koopman eigenfunctions and invariant subspaces. We show how the accuracy of the approximation can be tuned using a design parameter.

II. Preliminaries

In this section[1], we review basic concepts on the Koopman operator and Extended Dynamic Mode Decomposition.

A. Koopman Operator

Here, we introduce the (discrete-time) Koopman operator and its spectral properties following [6]. Consider a nonlinear,

[1]We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{C}$, the sets of natural, nonnegative integer, real, positive real, and complex numbers respectively. For a matrix $A \in \mathbb{C}^{m\times n}$, we denote the sets comprised of its rows by $\mathrm{rows}(A)$, its columns by $\mathrm{cols}(A)$, the number of its rows by $\sharp\mathrm{rows}(A)$, and the number of its columns by $\sharp\mathrm{cols}(A)$, 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. For $1 \leq i < k \leq m$, we denote by $A_{i:k}$ the matrix formed with the ith to kth rows of A. Moreover, $A_{i,j}$ denotes the ijth element of A. For a square nonsingular matrix B, we denote its inverse by B^{-1}. 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. The angle between vectors $v, w \in \mathbb{R}^n$ is $\angle(v, w)$. Given $v_1, \ldots, v_k \in \mathbb{C}^n$, $\mathrm{span}\{v_1, \ldots, v_k\}$ represents the set comprised of all linear combinations $c_1v_1 + \cdots + c_nv_n$, with $c_1, \ldots, c_n \in \mathbb{C}$. We use j to denote the imaginary unit (the solution of $x^2 + 1 = 0$). 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 $\|v\|_2 := \sqrt{v^Hv}$. Given a set A, we denote its complement by A^c. Given sets A and B, $A \subseteq B$ means that A is a subset of B. We denote by $A \cap B$ and $A \cup B$ the intersection and union of A and B, and set $A \setminus B := A \cap B^c$. Given a sequence of sets $\{A_i\}_{i=1}^{\infty}$, we denote its superior and inferior limits by $\limsup_{i\to\infty} A_i$ and $\liminf_{i\to\infty} A_i$, respectively. We refer to the set consisting of all continuous strictly increasing functions $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $\alpha(0) = 0$ by class-$\mathcal{K}$. Given $f : B \to A$ and $g : C \to B$, $f \circ g : C \to A$ denotes their composition.

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

$$x^+ = T(x). \tag{1}$$

The dynamics (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 (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 from $\mathcal{M}$ to $\mathbb{C}$ which is closed under composition with T, i.e.,

$$f \circ T \in \mathcal{F}, \quad \forall f \in \mathcal{F}. \tag{2}$$

The Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ associated with (1) is

$$\mathcal{K}(f) = f \circ T.$$

A closer look at the definition of the Koopman operator shows that it advances the observables in time, i.e., for $g = \mathcal{K}(f)$ then

$$g(x) = f \circ T(x) = f(x^+), \quad \forall x \in \mathcal{M}. \tag{3}$$

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 $c_1, c_2 \in \mathbb{C}$,

$$\mathcal{K}(c_1 f_1 + c_2 f_2) = c_1 \mathcal{K}(f_1) + c_2 \mathcal{K}(f_2). \tag{4}$$

Assuming $\mathcal{F}$ contains the functions describing the states of the system, $g_i(x) = x_i$ with $i \in \{1, \dots, n\}$, 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).

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

$$\mathcal{K}(\phi) = \lambda \phi. \tag{5}$$

The combination of (3) and (5) leads to a significant property of the Koopman operator: the linear evolution of its eigenfunctions in time. Formally, given an eigenfunction ϕ,

$$\phi(x^+) = (\phi \circ T)(x) = \mathcal{K}(\phi)(x) = \lambda \phi(x). \tag{6}$$

The linear evolution of eigenfunctions, together with linearity (4), enables us to use spectral properties to analyze the nonlinear system (1). Given a set of eigenpairs $\{(\lambda_i, \phi_i)\}_{i=1}^{N_k}$ such that $\mathcal{K}(\phi_i) = \lambda_i \phi_i, i \in \{1, \dots, N_k\}$, 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

$$f(x(k)) = \sum_{i=1}^{N_k} c_i \lambda_i^k \, \phi_i(x(0)), \quad \forall k \in \mathbb{N}_0. \tag{7}$$

The constants $\{c_i\}_{i=1}^{N_k}$ are called *Koopman modes*. It is important to note that one might need to use $N_k = \infty$ 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.

B. Extended Dynamic Mode Decomposition

Our exposition here mainly follows [35]. As mentioned earlier, despite its linearly, the infinite-dimensional nature of the Koopman operator obstructs the use of efficient linear algebraic methods. One natural way to overcome this problem is finding finite-dimensional approximations for it. Extended Dynamic Mode Decomposition (EDMD) is a popular data-driven method to perform this task that lifts data snapshots acquired from the dynamical system to a higher-dimensional space using a predefined dictionary of functions. The projection of the action of the operator on the span of the dictionary can then be found by solving a least-squares problem.

Formally, let $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^{1 \times N_d}$ be a dictionary of N_d functions in $\mathcal{F}$ with $\mathcal{D}(x) = [d_1(x), \dots, d_{N_d}(x)]$. Moreover, let $X, Y \in \mathbb{R}^{N \times n}$ be matrices comprised of N data snapshots such that $y_i = T(x_i)$ for $i \in \{1, \dots, N\}$, where x_i^T and y_i^T are ith rows of X and Y, respectively. For convenience, we define the action of the dictionary on a matrix as

$$\mathcal{D}(X) := [\mathcal{D}(x_1)^T, \dots, \mathcal{D}(x_N)^T]^T.$$

The EDMD method approximates the projection of the Koopman operator by finding the matrix that best explains the data over the dictionary, i.e.,

$$\underset{K}{\text{minimize}} \|\mathcal{D}(Y) - \mathcal{D}(X)K\|_F^2,$$

which yields the closed-form solution

$$K^* = \text{EDMD}(\mathcal{D}, X, Y) := \mathcal{D}(X)^\dagger \mathcal{D}(Y). \tag{8}$$

Note that the solution depends on the choice of dictionary. If the dictionary spans a Koopman-invariant subspace, then $\|\mathcal{D}(Y) - \mathcal{D}(X)K^*\|_F^2 = 0$ and K^* fully captures the evolution of functions in $\operatorname{span}(\mathcal{D}(x))$. Otherwise, EDMD loses some information about the dynamics.

III. Problem Statement

As described in Section II-B, the EDMD method loses information about the dynamical system when the employed dictionary does not span a Koopman-invariant subspace. As a result, in such cases, the EDMD approximation is not appropriate for long term prediction of the state evolution. Motivated by this observation, our goal is to find the maximal Koopman-invariant subspace and Koopman eigenfunctions in the span of a given dictionary.

Formally, given the dynamical system (1) defined by $T : \mathcal{M} \to \mathcal{M}$, data matrices X and Y comprised of N data snapshots, and an arbitrary dictionary of functions D, our main goal is two-fold:

(a) find all the Koopman eigenfunctions in $\operatorname{span}(D(x))$;
(b) find a basis for the maximal Koopman-invariant subspace in $\operatorname{span}(D(x))$.

Note that (a) and (b) are closely related. The eigenfunctions found by solving (a) span Koopman-invariant subspaces. Those invariant subspaces however might not be maximal. This mild difference between (a) and (b) 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.

Throughout the paper, we use the following assumption regarding the dictionary snapshots.

Assumption 3.1: (Full Column Rank Dictionary Matrices): The matrices $D(X)$ and $D(Y)$ have full column rank. □

Assumption 3.1 is reasonable: in order to hold, the dictionary functions must be linearly independent, i.e., the functions must form a basis for $\mathrm{span}(D(x))$. Moreover, the assumption requires the set of initial conditions $\mathrm{rows}(X)$ to be diverse enough to capture important characteristics of the dynamics. Our treatment here relies on EDMD, 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.

IV. EDMD and Koopman Eigenfunctions

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

$$K_f = \mathrm{EDMD}(D, X, Y), \quad K_b = \mathrm{EDMD}(D, Y, X).$$

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 4.1: (EDMD Captures the Koopman Eigenfunctions in the Span of the Dictionary): Suppose Assumption 3.1 holds. Let $f(x) = D(x)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$ and all $x \in \mathcal{M}$.

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

(b) 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: (a) 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.

(b) 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 (a). ■

Despite its simplicity, this result provides significant insight into the EDMD method. Lemma 4.1 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 [35], [45]. We note that Lemma 4.1 is a generalization of [27, 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. Its proof relies on Lemma 4.1 and is presented in the online version [46].

Proposition 4.2: (EDMD Accurately Predicts Evolution of any Linear Combination of Eigenfunctions on System's Trajectories): Let $f(x) = D(x)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$ and all $x \in \mathcal{M}$. Assume f is in the span of eigenfunctions $\{\phi_i\}_{i=1}^m \subset \mathrm{span}(D)$ with corresponding eigenvalues $\{\lambda_i\}_{i=1}^m \subset \mathbb{C}$. Then, given any trajectory $\{x(j)\}_{j=0}^\infty$ of (1),

$$f(x(j)) = D(x(0))K_f^j v, \ \forall j \in \mathbb{N}_0. \tag{9}$$

Lemma 4.1 provides a necessary condition for the identification of Koopman eigenfunctions. This condition however is not sufficient, see e.g. [1, 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 4.3: (Identification of Linear Evolutions by Forward and Backward EDMD): Suppose Assumption 3.1 holds. Let $f(x) = D(x)v$ for some $v \in \mathbb{C}^{N_d} \setminus \{0\}$. Then $f(y_i) = \lambda f(x_i)$ for some $\lambda \in \mathbb{C} \setminus \{0\}$ and for all $i \in \{1, \dots, \sharp\mathrm{rows}(X)\}$ 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, 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,

$$\lambda D(X)^T D(X)v = D(X)^T D(Y)v, \tag{11a}$$
$$\lambda^{-1} D(Y)^T D(Y)v = D(Y)^T D(X)v. \tag{11b}$$

By multiplying (11a) from the left by v^H and using (11b),

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

which implies

$$|\lambda|^2 \|D(X)v\|_2^2 = \|D(Y)v\|_2^2. \tag{12}$$

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

$$D(Y)v = cD(X)v + w, \tag{13}$$

with $v^H D(X)^T w = 0$. Substituting (13) into (11a) 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, we deduce that $c = \lambda$. Substituting the value of c in (13), 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 (12), 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 $f(x) = D(x)v$.

($\Rightarrow$): Based on Lemma 4.1(a), 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 $i \in \{1, \dots, \sharp\text{rows}(X)\}$ and, consequently, using Lemma 4.1(a) once again, we have $K_b v = \lambda^{-1} v$, concluding the proof. ∎

If the function f satisfies the conditions provided by Theorem 4.3, then $f(x^+) = \lambda f(x)$ for all $x \in \text{rows}(X)$. However, Theorem 4.3 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 next infinite sampling and make an assumption about its density.

Assumption 4.4: (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 $N \in \mathbb{N}$,

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

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

Assumption 4.4 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 4.4 holds for most standard random samplings.

Noting that our methods presented later require Assumption 3.1 to hold, we provide a definition for dictionary matrices acquired from infinite sampling.

Definition 4.5: (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 (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 $R = \min\{M \in \mathbb{N} \,|\, \text{rank}(D(X_{1:M})) = \text{rank}(D(Y_{1:M})) = N_d\}$ exists (R is called *richness constant*).

In Definition 4.5, if

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

then based on the well-ordering principle, see e.g. [47, 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 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 4.6: (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 $v \in \mathbb{C}^{N_d} \backslash \{0\}$ and $\lambda \in \mathbb{C} \backslash \{0\}$, let $f(x) = D(x)v$. Then,

(a) 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 $N \geq R$;

(b) Conversely, and assuming the dictionary functions are continuous and Assumption 4.4 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: (a) Since f is a Koopman eigenfunction, for every $i \in \mathbb{N}$ we have $f(y_i) = \lambda f(x_i)$. 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 4.3.

(b) Based on Theorem 4.3, we deduce that, for every $N \geq R$

$$f(y_i) = \lambda f(x_i) v, \ \forall i \in \{1, \dots, N\}, \tag{14}$$

where x_i^T and y_i^T are the ith rows of $X_{1:N}$ and $Y_{1:N}$ respectively. Now, define $h(x) := f \circ T(x) - \lambda f(x)$. 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 (14) 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 4.4, 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. ∎

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 [31], [32]. To our knowledge, the use of this technique here for the identification of Koopman eigenfunctions and invariant subspaces is novel. Moreover, unlike [48, Algorithm 1], the methods proposed here do not require access to the system's multi-step trajectories. Theorems 4.3 and 4.6 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.

V. Identification of Koopman-Invariant Subspaces via Symmetric Subspace Decomposition (SSD)

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 III, 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 $\mathrm{span}(D)$. Since $\mathrm{span}(\tilde{D})$ is invariant, we have $\mathcal{R}(\tilde{D}\circ T) = \mathcal{R}(\tilde{D})$. This equality gets reflected in the data, i.e., given snapshot matrices X and Y,

$$\mathcal{R}(\tilde{D}(Y)) = \mathcal{R}(\tilde{D}(X)). \tag{15}$$

Moreover, since the elements of $\tilde{D}$ are in the span of D, there exists a full column rank matrix C such that $\tilde{D}(x) = D(x)C$, for all $x \in \mathcal{M}$. Thus from (15),

$$\mathcal{R}(D(Y)C) = \mathcal{R}(D(X)C). \tag{16}$$

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 (16) holds. To solve this problem, we propose the Symmetric Subspace Decomposition (SSD) method. The SSD algorithm relies on the fact, from (15), 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[2].

Algorithm 1 Symmetric Subspace Decomposition

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

A. 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 5.1: (Properties of SSD Output): Suppose Assumption 3.1 holds. For matrices $D(X), D(Y)$, let $C^{\mathrm{SSD}} = \mathrm{SSD}(D(X), D(Y))$. The SSD algorithm has the following properties:

(a) it stops after at most N_d iterations;
(b) the matrix C^{SSD} is either 0 or has full column rank, and satisfies $\mathcal{R}(D(X)C^{\mathrm{SSD}}) = \mathcal{R}(D(Y)C^{\mathrm{SSD}})$;
(c) the subspace $\mathcal{R}(D(X)C^{\mathrm{SSD}})$ is maximal, in the sense that, for any matrix E with $\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E)$, we have $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C^{\mathrm{SSD}})$ and $\mathcal{R}(E) \subseteq \mathcal{R}(C^{\mathrm{SSD}})$;
(d) $\mathcal{R}\big(\mathrm{SSD}(D(X), D(Y))\big) = \mathcal{R}\big(\mathrm{SSD}(D(Y), D(X))\big)$.

Proof: (a) 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, A_1 and B_1 have full column rank. Now, by using Lemma A.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 one can deduce that $A_{k+1} = A_1 Z_1^A \ldots Z_k^A$, $B_{k+1} = B_1 Z_1^A \ldots Z_k^A$ have full column rank since they are product of matrices with full column rank. Using Lemma A.1, one can conclude that Z_{k+1}^A and Z_{k+1}^B have full column rank.

Consequently, we have $\sharp\mathrm{rows}(Z_i^A) \geq \sharp\mathrm{cols}(Z_i^A)$. Hence, Step 9 of the SSD algorithm implies that the algorithm can only move to the next iteration if $\sharp\mathrm{rows}(Z_i^A) > \sharp\mathrm{cols}(Z_i^A)$, 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.

(b) The $C^{\mathrm{SSD}} = 0$ 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 $A_k = D(X)C^{\mathrm{SSD}}$ and $B_k = D(Y)C^{\mathrm{SSD}}$. Hence, $\mathcal{R}(D(X)C^{\mathrm{SSD}}) = \mathcal{R}(D(Y)C^{\mathrm{SSD}})$. Moreover, $C^{\mathrm{SSD}} = Z_1^A \cdots Z_{k-1}^A$ and considering the fact that $Z_1^A, \ldots, Z_{k-1}^A$ have full column rank, one can deduce that C^{SSD} has full column rank.

(c) 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

$$\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_i) \cap \mathcal{R}(B_i). \tag{17}$$

[2]The function $\mathrm{null}([A_i, B_i])$ 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.

Using Lemma A.1, one can derive $\mathcal{R}(A_i Z_i^A) = \mathcal{R}(A_i) \cap \mathcal{R}(B_i)$. Combining this with (17), we get

$$\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_i Z_i^A) = \mathcal{R}\big(D(X)Z_1^A \cdots Z_i^A\big). \quad (18)$$

Using (18) with Lemma A.2 one can derive $\mathcal{R}(E) \subseteq \mathcal{R}(Z_1^A \cdots Z_i^A)$. Using Lemma A.2 once again, we get

$$\mathcal{R}(D(X)E) = \mathcal{R}(D(Y)E) \subseteq \mathcal{R}\big(D(Y)Z_1^A \cdots Z_i^A\big). \quad (19)$$

Definition of A_{i+1}, B_{i+1} along with (18) and (19) 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 $C^{\mathrm{SSD}} = 0$, we have $\mathcal{R}(A_k) \cap \mathcal{R}(B_k) = \{0\}$, which means that $E = 0$ and $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C^{\mathrm{SSD}})$. In the case that $C^{\mathrm{SSD}} \neq 0$, using the fact that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(A_k) \cap \mathcal{R}(B_k)$, $C^{\mathrm{SSD}} = Z_1^A \cdots Z_{k-1}^A$, and $\mathcal{R}(D(X)C^{\mathrm{SSD}}) = \mathcal{R}(D(Y)C^{\mathrm{SSD}})$, one can deduce that $\mathcal{R}(D(X)E) \subseteq \mathcal{R}(D(X)C^{\mathrm{SSD}})$. Moreover, using Assumption 3.1 and Lemma A.2 one can write $\mathcal{R}(E) \subseteq \mathcal{R}(C^{\mathrm{SSD}})$.

(d) For convenience, let $E^{\mathrm{SSD}} = \mathrm{SSD}(D(Y), D(X))$. Based on the definition of C^{SSD} and E^{SSD}, one can write

$$\mathcal{R}(D(X)C^{\mathrm{SSD}}) = \mathcal{R}(D(Y)C^{\mathrm{SSD}})$$
$$\mathcal{R}(D(X)E^{\mathrm{SSD}}) = \mathcal{R}(D(Y)E^{\mathrm{SSD}})$$

These equations in conjunction with the maximality of $\mathcal{R}(C^{\mathrm{SSD}})$ from part (c) imply $\mathcal{R}(E^{\mathrm{SSD}}) \subseteq \mathcal{R}(C^{\mathrm{SSD}})$. Using a similar argument, invoking the maximality of $\mathcal{R}(E^{\mathrm{SSD}})$, we have $\mathcal{R}(C^{\mathrm{SSD}}) \subseteq \mathcal{R}(E^{\mathrm{SSD}})$, concluding the proof. ■

Remark 5.2: (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., [49]. Since, based on Theorem 5.1(a), 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 5.3: (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

$$\mathrm{rows}\big([D(X), D(Y)]\big) \subseteq \mathrm{rows}\big([D(\hat{X}), D(\hat{Y})]\big), \quad (20)$$

and for which Assumption 3.1 holds. Then

$$\mathcal{R}(\mathrm{SSD}([D(\hat{X}), D(\hat{Y})])) \subseteq \mathcal{R}(\mathrm{SSD}(D(X), D(Y))).$$

Proof: We use the shorthand notation $\hat{C} = \mathrm{SSD}([D(\hat{X}), D(\hat{Y})])$ and $C = \mathrm{SSD}(D(X), D(Y))$. From (20), we deduce that there exists a matrix E with $\mathrm{rows}(E) \subseteq \mathrm{rows}(I_{\sharp \mathrm{rows}(\hat{X})})$ such that

$$ED(\hat{X}) = D(X),\ ED(\hat{Y}) = D(Y). \quad (21)$$

Moreover, based on the definition of $\hat{C}$ and Theorem 5.1(b), 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 (21) 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 5.1(c)) implies $\mathcal{R}(\hat{C}) \subseteq \mathcal{R}(C)$. ■

Remark 5.4: (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

$$\sum_{j=k_i}^{l_i} \sigma_j^2 \leq \epsilon\big(\sum_{j=1}^{l_i} \sigma_j^2\big). \quad (22)$$

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

$$\|[A_i, B_i] - [\hat{A}_i, \hat{B}_i]\|_F^2 \leq \epsilon \|[A_i, B_i]\|_F^2. \quad (23)$$

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. [50]. □

B. 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

$$\tilde{D}(x) := D(x)C^{\mathrm{SSD}}. \quad (24)$$

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

$$\begin{aligned} K^{\mathrm{SSD}} &= \mathrm{EDMD}(\tilde{D}, X, Y) = \tilde{D}(X)^{\dagger}\tilde{D}(Y) \\ &= \big(D(X)C^{\mathrm{SSD}}\big)^{\dagger}\big(D(Y)C^{\mathrm{SSD}}\big). \end{aligned} \quad (25)$$

Based on Theorem 5.1(b), we have

$$\mathcal{R}(\tilde{D}(X)) = \mathcal{R}(\tilde{D}(Y)). \quad (26)$$

Moreover, $\tilde{D}(X)$ and $\tilde{D}(Y)$ have full column rank as a result of Assumption 3.1 and Theorem 5.1(b). Consequently, K^{SSD} is a (unique) nonsingular matrix satisfying

$$\tilde{D}(Y) = \tilde{D}(X)K^{\mathrm{SSD}}. \quad (27)$$

Interestingly, equation (27) implies that the residual error of EDMD, $\|\tilde{D}(Y) - \tilde{D}(X)K^{\mathrm{SSD}}\|_F$, is equal to zero. Based on (26), 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 $D(\hat{X}), D(\hat{Y})$ such that

$$\mathrm{rows}[D(\hat{X}), D(\hat{Y})] \subseteq \mathrm{rows}[D(X), D(Y)].$$

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

Theorem 5.5: (Identification of Linear Evolutions using the SSD Algorithm): Suppose that Assumption 3.1 holds. Let $C^{\mathrm{SSD}} = \mathrm{SSD}(D(X), D(Y)) \neq 0$, $K^{\mathrm{SSD}} = \big(D(X)C^{\mathrm{SSD}}\big)^{\dagger}\big(D(Y)C^{\mathrm{SSD}}\big)$, and $f(x) \in \mathrm{span}(D(x))$ denoted as $f(x) = D(x)v$ with $v \in \mathbb{C}^{N_d} \setminus \{0\}$. Then $f(y_i) = \lambda f(x_i)$ 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, and considering the fact that C^{SSD} has full column rank (Theorem 5.1(b)), one can use (24)-(27) 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

$$D(Y)v = \lambda D(X)v. \tag{28}$$

Consider first the case when $v \in \mathbb{R}^{N_d}$. Then using (28), we deduce $\mathcal{R}(D(X)v) = \mathcal{R}(D(Y)v)$. The maximality of C^{SSD} (Theorem 5.1(c)) 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 (28) 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 (28) and write $D(Y)\bar{v} = \bar{\lambda}D(X)\bar{v}$. This, together with (28), implies

$$D(Y)E = D(X)E\Lambda, \tag{29}$$

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 5.1(c), 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 (29) 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. ■

Using Theorem 5.5, 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 IV.

Corollary 5.6: (Equivalence of Forward-Backward EDMD and SSD in the Identification of Linear Evolutions): Suppose that Assumption 3.1 holds. Let $K_f = \mathrm{EDMD}(D, X, Y)$, $K_b = \mathrm{EDMD}(D, Y, X)$, $C^{\mathrm{SSD}} = \mathrm{SSD}(D(X), D(Y)) \neq 0$ and $K^{\mathrm{SSD}} = \big(D(X)C^{\mathrm{SSD}}\big)^{\dagger}\big(D(Y)C^{\mathrm{SSD}}\big)$. 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 $v = C^{\mathrm{SSD}}w$ and $K^{\mathrm{SSD}}w = \lambda w$.

The proof of this result is a consequence of Theorems 4.3 and 5.5. 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 5.7: (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 $C_N^{\mathrm{SSD}} = \mathrm{SSD}(D(X_{1:N}), D(Y_{1:N})) \neq 0$, and $K_N^{\mathrm{SSD}} = \big(D(X_{1:N})C_N^{\mathrm{SSD}}\big)^{\dagger}\big(D(Y_{1:N})C_N^{\mathrm{SSD}}\big)$. Given $v \in \mathbb{C}^{N_d} \setminus \{0\}$ and $\lambda \in \mathbb{C} \setminus \{0\}$, let $f(x) = D(x)v$. Then,

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

(b) Conversely, and assuming the dictionary functions are continuous and Assumption 4.4 holds, if $v \in \mathcal{R}(C_N^{\mathrm{SSD}})$ and there exists w_N such that $v = C_N^{\mathrm{SSD}}w_N$ and $K_N^{\mathrm{SSD}}w_N = \lambda w_N$ for every $N \geq R$, then f is an eigenfunction of the Koopman operator with probability 1.

This result is a consequence of Theorem 4.6 and Corollary 5.6. Theorem 5.7 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 5.8: (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 4.4 holds. Let the columns of C_∞^{SSD} form a basis for $\lim_{N\to\infty} \mathcal{R}(C_N^{\mathrm{SSD}})$, i.e.,

$$\mathcal{R}(C_\infty^{\mathrm{SSD}}) = \lim_{N\to\infty} \mathcal{R}(C_N^{\mathrm{SSD}}) = \bigcap_{N=R}^{\infty} \mathcal{R}(C_N^{\mathrm{SSD}}). \tag{30}$$

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

Proof: If $C_\infty^{\mathrm{SSD}} = 0$, considering the fact that for all $N \geq R$, $\mathcal{R}(C_{N+1}^{\mathrm{SSD}}) \subseteq \mathcal{R}(C_N^{\mathrm{SSD}})$ (Lemma 5.3), one deduces that there exists $m \in \mathbb{N}$ such that for all $i \geq m$, $C_i^{\mathrm{SSD}} = 0$.

Hence based on Theorem 5.1(c), 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 $D(x)C_\infty^{\mathrm{SSD}}$.

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

$$\mathcal{R}(D(X_{1:N})C_\infty^{\mathrm{SSD}}) = \mathcal{R}(D(Y_{1:N})C_\infty^{\mathrm{SSD}}), \quad \forall N \geq R. \quad (31)$$

Considering (30) and the fact that for all $N \geq R$, $\mathcal{R}(C_{N+1}^{\mathrm{SSD}}) \subseteq \mathcal{R}(C_N^{\mathrm{SSD}})$, we can write for all $N \geq R$

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

Invoking Lemma A.4, we have for all $N \geq R$,

$$\mathcal{R}(D(X_{1:N})C_\infty^{\mathrm{SSD}}) = \bigcap_{i=N}^{\infty} \mathcal{R}(D(X_{1:N})C_i^{\mathrm{SSD}}), \quad (32a)$$

$$\mathcal{R}(D(Y_{1:N})C_\infty^{\mathrm{SSD}}) = \bigcap_{i=N}^{\infty} \mathcal{R}(D(Y_{1:N})C_i^{\mathrm{SSD}}). \quad (32b)$$

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

$$\mathcal{R}(D(X_{1:N})C_i^{\mathrm{SSD}}) = \mathcal{R}(D(Y_{1:N})C_i^{\mathrm{SSD}}), \quad \forall i \geq N. \quad (33)$$

The combination of (32) and (33) yields (31). 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_∞^{SSD} has full column rank, there exists a *unique* nonsingular square matrix $K_\infty^{\mathrm{SSD}} \in \mathbb{R}^{\sharp\mathrm{cols}(C_\infty^{\mathrm{SSD}}) \times \sharp\mathrm{cols}(C_\infty^{\mathrm{SSD}})}$ such that

$$D(X_{1:N})C_\infty^{\mathrm{SSD}}K_\infty^{\mathrm{SSD}} = D(Y_{1:N})C_\infty^{\mathrm{SSD}}, \quad \forall N \geq R. \quad (34)$$

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

$$g(x) := D(x)C_\infty^{\mathrm{SSD}}K_\infty^{\mathrm{SSD}}v. \quad (35)$$

We show that $g = f \circ T = \mathcal{K}(f)$ almost surely. Define the function $h := g - f \circ T$. Also, let $S_\infty = \bigcup_{N=R}^{\infty} S_N$ be the set of initial conditions. Based on (34), (35), 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 4.4), we deduce $h \equiv 0$ on $\mathcal{M}$ almost surely. Therefore, $g = \mathcal{K}(f) = f \circ T$ with probability 1. Noting that $g(x) \in \mathrm{span}(D(x)C_\infty^{\mathrm{SSD}})$, we have proven that $\mathrm{span}(D(x)C_\infty^{\mathrm{SSD}})$ is Koopman invariant almost surely.

Finally, we prove the maximality of $\mathrm{span}(D(x)C_\infty^{\mathrm{SSD}})$. Let $\mathcal{L}$ be a Koopman-invariant subspace in $\mathrm{span}(D(x))$. Then there exists a full column rank matrix E such that $\mathcal{L} = \mathrm{span}(D(x)E)$. 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 5.1(c), we have $\mathcal{R}(E) \subseteq \mathcal{R}(C_N^{\mathrm{SSD}})$, for all $N \geq R$, and hence $\mathcal{R}(E) \subseteq \mathcal{R}(C_\infty^{\mathrm{SSD}})$. Therefore, by Lemma A.2, we have $\mathcal{L} = \mathcal{R}(D(x)E) \subseteq \mathcal{R}(D(x)C_\infty^{\mathrm{SSD}})$, which completes the proof. ■

Remark 5.9: (Generalized Koopman Eigenfunctions): One can also extend the above discussion for generalized Koopman eigenfunctions (see e.g. [6, Remark 11]). Given a generalized eigenvector w of K^{SSD}, the corresponding generalized Koopman eigenfunction is $\phi(x) = D(x)C^{\mathrm{SSD}}w$. □

VI. 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 maybe 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

Initialization
$D_S^X(1) \leftarrow \begin{bmatrix} D(X_{1:S}) \\ D(x_{S+1}) \end{bmatrix}$, $D_S^Y(1) \leftarrow \begin{bmatrix} D(Y_{1:S}) \\ D(y_{S+1}) \end{bmatrix}$
$i \leftarrow 1$, $A_1 \leftarrow D_S^X(1)$, $B_1 \leftarrow D_S^Y(1)$, $C_0 \leftarrow I_{N_d}$
while 1 **do**
  if $C_{i-1} = 0$ **then**
    $C_i \leftarrow 0$ ▷ The basis does not exist
    return C_i
    break
  end if
  $F_i \leftarrow \mathrm{SSD}(A_i, B_i)$
  if $F_i = 0$ **then**
    $C_i \leftarrow 0$ ▷ The basis does not exist
    return C_i
    break
  end if
  if $\sharp\mathrm{rows}(F_i) > \sharp\mathrm{cols}(F_i)$ **then**
    $C_i \leftarrow \mathrm{basis}(\mathcal{R}(C_{i-1}F_i))$ ▷ Subspace reduction
  else
    $C_i \leftarrow C_{i-1}$ ▷ No change
  end if
  return C_i
  $i \leftarrow i + 1$
  ▽ Replacing the last data snapshot with the new one
  $D_S^X(i) = \begin{bmatrix} D(X_{1:S}) \\ D(x_{S+i}) \end{bmatrix}$, $D_S^Y(i) = \begin{bmatrix} D(Y_{1:S}) \\ D(y_{S+i}) \end{bmatrix}$
  ▽ Calculating the reduced dictionary snapshots
  $A_i \leftarrow D_S^X(i)C_{i-1}$, $B_i \leftarrow D_S^Y(i)C_{i-1}$
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 5.3), i.e., by adding more data the dimension of the dictionary does not increase. Since the SSD algorithm relies on Assumption 3.1, we make the following assumption on the signature snapshots and the original dictionary.

Assumption 6.1: (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. □

For a finite number of data snapshots, Assumption 6.1 is equivalent to Assumption 3.1. For an infinite sampling, Assumption 6.1 holds for a R-rich sequence of snapshot matrices. The next result discusses the basic properties of the SSSD output at each iteration.

Proposition 6.2: (Properties of SSSD Output): Suppose Assumption 6.1 holds. For $i \in \mathbb{N}$, let C_i denote the output of the SSSD algorithm at the ith iteration. Then, for all $i \in \mathbb{N}$,

(a) C_i has full column rank or is equal to zero;
(b) $\mathcal{R}(C_i) \subseteq \mathcal{R}(C_{i-1})$;
(c) $\mathcal{R}(D_S^X(i)C_i) = \mathcal{R}(D_S^Y(i)C_i)$.

Proof: (a) 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 5.1(b), 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 (b). 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 5.1(b), 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 A.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 (c). 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 5.1(b)). 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,

$$\mathcal{R}(C_i) = \mathcal{R}(C_{i-1}) = \mathcal{R}(C_{i-1}I_{\sharp\text{cols}(C_{i-1})}) = \mathcal{R}(C_{i-1}F_i). \tag{36}$$

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

$$\mathcal{R}(C_i) = \mathcal{R}(C_{i-1}F_i). \tag{37}$$

Also, based on definition of F_i at Step 10, Theorem 5.1(b), 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 (36) upon execution of Step 19 and (37) 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 6.3: (Equivalence of SSD and SSSD): Suppose Assumption 6.1 holds. For $i \in \mathbb{N}$, let C_i denote the output of the SSSD algorithm at the ith iteration and let $C_i^{\text{SSD}} = \text{SSD}\big(D(X_{1:S+i}), D(Y_{1:S+i})\big)$. Then,

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

Proof: Inclusion $\mathcal{R}(C_i^{\text{SSD}}) \subseteq \mathcal{R}(C_i)$ for all $i \in \mathbb{N}$: We reason by induction. Note that in the SSSD algorithm, for $i = 1$ we have $F_1 = C_1^{\text{SSD}}$. As a result, if $F_1 = 0$ then based on Step 12, $C_1 = C_1^{\text{SSD}} = 0$. If the SSSD algorithm executes Step 17, then using the fact that $C_0 = I_{N_d}$, one can write $\mathcal{R}(C_1) = \mathcal{R}(C_1^{\text{SSD}})$. Moreover, if the SSSD algorithm executes Step 19, based on Step 16 and Theorem 5.1(b), one can deduce that $\mathcal{R}(C_1^{\text{SSD}}) = \mathcal{R}(C_1) = \mathcal{R}(F_1) = \mathcal{R}(I_{N_d})$. Consequently, in all cases

$$\mathcal{R}(C_1^{\text{SSD}}) = \mathcal{R}(C_1). \tag{38}$$

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

$$\mathcal{R}(C_k^{\text{SSD}}) \subseteq \mathcal{R}(C_k). \tag{39}$$

We need to show that $\mathcal{R}(C_{k+1}^{\text{SSD}}) \subseteq \mathcal{R}(C_{k+1})$. If $C_{k+1}^{\text{SSD}} = 0$ then the proof follows. Now assume that $C_{k+1}^{\text{SSD}} \neq 0$ and has full column rank based on Theorem 5.1(b). By Lemma 5.3, we have

$$\mathcal{R}(C_{k+1}^{\text{SSD}}) \subseteq \mathcal{R}(C_k^{\text{SSD}}). \tag{40}$$

Using (39) and (40), one can deduce $\mathcal{R}(C_{k+1}^{\text{SSD}}) \subseteq \mathcal{R}(C_k)$. Consequently, based on the fact that $C_{k+1}^{\text{SSD}} \neq 0$, we have $C_k \neq 0$ and hence has full column rank based on Proposition 6.2(a). Moreover, there exists a full column-rank matrix E_k such that

$$C_{k+1}^{\text{SSD}} = C_k E_k. \tag{41}$$

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 (41) and considering the fact that $\mathcal{R}(E_k) \subseteq \mathcal{R}(I_{\sharp\text{cols}(C_k)})$ and the fact that C_k has full column rank, one can use Lemma A.2 and conclude

$$\mathcal{R}(C_{k+1}^{\text{SSD}}) = \mathcal{R}(C_k E_k) \subseteq \mathcal{R}(C_k) = \mathcal{R}(C_{k+1}). \tag{42}$$

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

$$\mathcal{R}(C_{k+1}) = \mathcal{R}(C_k F_{k+1}). \tag{43}$$

Also, based on definition of C_{k+1}^{SSD} and Theorem 5.1(b), one can write

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

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

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

Now, using (41) we have $\mathcal{R}(D_S^X(k+1)C_kE_k) = \mathcal{R}(D_S^Y(k+1)C_kE_k)$. 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 5.1(c) to write $\mathcal{R}(E_k) \subseteq \mathcal{R}(F_{k+1})$. Since C_k has full column rank, we use (41), (43), and Lemma A.2 to write

$$\mathcal{R}(C_{k+1}^{\mathrm{SSD}}) = \mathcal{R}(C_kE_k) \subseteq \mathcal{R}(C_kF_{k+1}) = \mathcal{R}(C_{k+1}). \quad (44)$$

In both cases, equations (42) and (44) conclude the induction.

Inclusion $\mathcal{R}(C_i) \subseteq \mathcal{R}(C_i^{\mathrm{SSD}})$ for all $i \in \mathbb{N}$: We reason by induction too. Using (38), we have $\mathcal{R}(C_1) \subseteq \mathcal{R}(C_1^{\mathrm{SSD}})$. Now, suppose that

$$\mathcal{R}(C_k) \subseteq \mathcal{R}(C_k^{\mathrm{SSD}}). \quad (45)$$

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 (45), Proposition 6.2(a), and Theorem 5.1(b), we deduce that C_{k+1} and C_{k+1}^{SSD} have full column rank.

The first part of the proof and (45) imply that $\mathcal{R}(C_k) = \mathcal{R}(C_k^{\mathrm{SSD}})$. Consequently, noting the fact that C_k^{SSD} is the output of the SSD algorithm with $D(X_{1:S+k})$ and $D(Y_{1:S+k})$, one can use Theorem 5.1(b) to write

$$\mathcal{R}\big(D(X_{1:S+k})C_k\big) = \mathcal{R}\big(D(Y_{1:S+k})C_k\big). \quad (46)$$

Moreover, based on Proposition 6.2(b), 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

$$C_{k+1} = C_kG_k. \quad (47)$$

Also, based on Proposition 6.2(c) at iteration $k+1$ of the SSSD algorithm

$$\mathcal{R}(D_S^X(k+1)C_{k+1}) = \mathcal{R}(D_S^Y(k+1)C_{k+1}). \quad (48)$$

Consequently, based on (47) and (48), 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 (46) and Lemma A.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 (47) 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

$$D(X_{1:S+k})C_{k+1}K^* = D(Y_{1:S+k})C_{k+1}. \quad (49)$$

Also, based on (48) 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

$$D_S^X(k+1)C_{k+1}K = D_S^Y(k+1)C_{k+1}. \quad (50)$$

Using the first S rows of (49) and (50), 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 (49) and (50) yields

$$\mathcal{R}\big(D(X_{1:S+k+1})C_{k+1}\big) = \mathcal{R}\big(D(Y_{1:S+k+1})C_{k+1}\big). \quad (51)$$

From (51), the definition of C_{k+1}^{SSD} and Theorem 5.1(c), we deduce $\mathcal{R}(C_{k+1}) \subseteq \mathcal{R}(C_{k+1}^{\mathrm{SSD}})$, concluding the proof. ■

Theorem 6.3 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 V are also valid for the output of the SSSD algorithm.

Remark 6.4: (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., [49]. 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 6.5: (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 5.2 and 6.4 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. □

VII. 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, 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[3].

Algorithm 3 approx-null(A, B, ϵ)

$\triangledown$ Singular value decomposition of $[A, B]$
$\{U, S, V\} \leftarrow \text{svd}([A, B])$ $\triangleright$ $USV^T = [A, B]$
$m \leftarrow \sharp\text{cols}(V)$ $\triangleright$ # of columns of V
$k_{\min} \leftarrow \left\{ \min_k \text{ s.t. } \left(\frac{\sum_{i=k}^{m} S_{i,i}^2}{\|S\|_F^2} \leq \epsilon^2 \wedge k > \sharp\text{cols}(A) \right) \right\}$
$\triangledown$ Choosing the right singular vectors corresponding to small singular values as the approximated null space
if $k_{\min} = \emptyset$ **then**
  return $\emptyset$
  break
else
  $Z \leftarrow (V_{k_{\min}:m}^T)^T$
end if
$\triangledown$ Make sure Assumption 3.1 holds for the output
while 1 **do**
  $\begin{bmatrix} Z^A \\ Z^B \end{bmatrix} \leftarrow Z$ $\triangleright$ $\sharp\text{rows}(Z^A) = \sharp\text{rows}(Z^B)$
  if $\text{rank}(Z^A) = \text{rank}(Z^B) = \sharp\text{cols}(Z)$ **then**
    return Z $\triangleright$ Basis for approximated null space
    break
  end if
  $\triangledown$ Reducing the space
  if $\sharp\text{cols}(Z) = 1$ **then**
    return $\emptyset$
    break
  else
    $Z \leftarrow (Z_{2:\sharp\text{cols}(Z)}^T)^T$ $\triangleright$ Removing the 1st column
  end if
end while

The next result studies the basic properties of Algorithm 3.

Proposition 7.1: (Properties of Algorithm 3): Let A and B be matrices of equal size, $\epsilon > 0$, and $Z = \text{approx-null}(A, B, \epsilon)$. Then,

(a) Algorithm 3 terminates in finite iterations;
(b) Z is either $\emptyset$ or has full column rank;
(c) 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: (a) 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 $k_{\min} > \sharp\text{cols}(A)$ and $m = \sharp\text{cols}(V) = \sharp\text{cols}(A) + \sharp\text{cols}(B)$ (cf. Step 2), we deduce

$$\sharp\text{cols}(Z_1) = m - k_{\min} \leq \sharp\text{cols}(B). \tag{52}$$

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 (52) 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 $\sharp\text{cols}(Z_j) \geq 0$.

(b) 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.

(c) 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 7.2: (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 $i \in \{\sharp\text{cols}(V) - \sharp\text{cols}(Z) + 1, \ldots, \sharp\text{cols}(V)\}$. 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,

(a) $\|W - \bar{W}\|_F \leq \epsilon \|W\|_F$;
(b) the columns of Z form a basis for the null space of $\bar{W}$;
(c) $\bar{A}Z^A = -\bar{B}Z^B$, where $Z = [(Z^A)^T, (Z^B)^T]^T$ and Z^A, Z^B have the same size.

Proof: (a) By construction we have $Z = V_{(\sharp\text{cols}(V)-\sharp\text{cols}(Z)+1):\sharp\text{cols}(V)}^T$, i.e., the columns of Z are the last $\sharp\text{cols}(Z)$ 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 $k_{\min} \leq \sharp\text{cols}(V) - \sharp\text{cols}(Z) + 1$, one can write

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

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

(b) 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 $\bar{W}$.

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

(c) Based on (b), $\bar{W}Z = \mathbf{0}$. Hence, $\bar{A}Z^A = -\bar{B}Z^B$. ∎

We formally define the *Approximated-SSD* algorithm 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.

For convenience, we denote the output of the Approximated SSD algorithm by $C_{\text{aprx}}^{\text{SSD}}$. Proposition 7.1 completely preserves the logical structure for the proof of Theorem 5.1(a) and, as a result, we deduce that the Approximated-SSD algorithm terminates in at most N_d iterations. Moreover, the $C_{\text{aprx}}^{\text{SSD}}$ matrix is zero or has full column rank, since the second part of the proof for Theorem 5.1(b) also holds for Approximated-SSD. If $C_{\text{aprx}}^{\text{SSD}} \neq 0$, one can define the reduced dictionary with $\tilde{N}_d = \sharp\text{cols}(C_{\text{aprx}}^{\text{SSD}})$ elements as

$$\tilde{D}_{\text{aprx}}(x) = D(x)C_{\text{aprx}}^{\text{SSD}}, \ \forall x \in \mathcal{M}. \tag{53}$$

We propose calculating the linear prediction matrix $K_{\text{aprx}}^{\text{SSD}}$ by solving the following total least squares (TLS) problem (see e.g. [51] for more information on TLS)

$$\underset{K,\Delta_1,\Delta_2}{\text{minimize}} \quad \|[\Delta_1, \Delta_2]\|_F \tag{54a}$$

$$\text{subject to} \quad \tilde{D}_{\text{aprx}}(Y) + \Delta_2 = (\tilde{D}_{\text{aprx}}(X) + \Delta_1)K. \tag{54b}$$

Even though TLS problems do not always have a solution, the next result shows that (54) 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 7.3: (Solution and Prediction Accuracy of (54)*):* 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

$$K_{\text{aprx}}^{\text{SSD}} = \bar{A}^\dagger \bar{B}, \tag{55a}$$

$$[\Delta_1^*, \Delta_2^*] = [\bar{A}, \bar{B}] - [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]. \tag{55b}$$

Then, $K_{\text{aprx}}^{\text{SSD}}$, Δ_1^*, Δ_2^* are the global solution of (54) and

$$\|[\Delta_1^*, \Delta_2^*]\|_F \leq \epsilon \|[\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]\|_F. \tag{56}$$

Proof: One can rewrite (54b) as

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

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 [52], 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 (55b) minimize the cost function in (54a). Next, we need to show that they also satisfy (54b) with $K_{\text{aprx}}^{\text{SSD}}$ defined in (55a).

Let t be the termination iteration of the Approximated-SSD algorithm. Since $C_{\text{aprx}}^{\text{SSD}} \neq 0$, 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 $[(Z_t^A)^T, (Z_t^B)^T] = \text{approx-null}(A_t, B_t, \epsilon)$. In addition, based on Proposition 7.1(c), Z_t^A and Z_t^B are nonsingular square matrices. Noting that by definition in the Approximated-SSD algorithm, $A_t = D(X)C_{\text{aprx}}^{\text{SSD}} = \tilde{D}_{\text{aprx}}(X)$ and $B_t = D(Y)C_{\text{aprx}}^{\text{SSD}} = \tilde{D}_{\text{aprx}}(Y)$, one can use Proposition 7.2(c) 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 $\bar{A}K_{\text{aprx}}^{\text{SSD}} = \bar{B}$, where $K_{\text{aprx}}^{\text{SSD}}$ is defined in (55a). As a result, $\Delta_1^*, \Delta_2^*, K_{\text{aprx}}^{\text{SSD}}$ satisfy the constraint (54b). Finally, the accuracy bound defined in (56) follows from Proposition 7.2(a) with $W = [\tilde{D}_{\text{aprx}}(X), \tilde{D}_{\text{aprx}}(Y)]$ and $\bar{W} = [\bar{A}, \bar{B}]$. ∎

Note that, unlike in the exact case (cf. Theorem 5.8), Theorem 7.3 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 $K_{\text{aprx}}^{\text{SSD}}$. 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.

VIII. Simulation Results

We illustrate the efficacy of the proposed methods in two examples.[4]

Example 8.1: (Unstable Discrete-time Polynomial System): Consider the nonlinear system

$$\begin{aligned} x_1^+ &= 1.1\, x_1 \\ x_2^+ &= 1.2\, x_2 + 0.1\, x_1^2 + 0.1, \end{aligned} \tag{57}$$

with state $x^T = [x_1, x_2]^T$. System (57) is actually an unstable Polyflow [53] which has a finite-dimensional Koopman-invariant subspace comprised of polynomials. 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 $\text{span}(D(x))$. 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. Similarly to the previous example, we use the strategy explained in Remark 5.4 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 $\text{span}(D(x))$. 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 (25). Moreover, based on Theorems 5.5 and 5.7, we use the eigendecomposition of K^{SSD} to find all the Koopman eigenfunctions associated with the system (57) in $\text{span}(D(x))$. Table I shows the identified eigenfunctions. One can use direct calculation to verify that the identified functions are the Koopman eigenfunctions associated with system (57). Note that since x_1 and x_2 are both

[4]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.

in the span of the identified Koopman-invariant subspace, one can fully characterize the behavior of the system using the eigenfunctions and (7) linearly or directly using the identified dictionary and K^{SSD}.

TABLE I: Identified eigenfunctions and eigenvalues of the Koopman operator associated with system (57).

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

Next, we evaluate the effectiveness of the original dictionary D and the dictionary $\tilde{D}$ 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 (57) with length M and initial condition x_0, let

$$E_{\text{relative}}(k) = \frac{\left\|\mathcal{D}(x(k)) - \mathcal{D}(x_0)K^k\right\|_2}{\|\mathcal{D}(x(k))\|_2} \times 100, \tag{58a}$$

$$E_{\text{angle}}(k) = \angle\big(\mathcal{D}(x(k)), \mathcal{D}(x_0)K^k\big), \tag{58b}$$

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 $E^{\text{Orig}}_{\text{relative}}$ and $E^{\text{Orig}}_{\text{angle}}$, and the errors associated to the SSD dictionary $\tilde{D}$, denoted $E^{\text{SSD}}_{\text{relative}}$ and $E^{\text{SSD}}_{\text{angle}}$. Figure 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 $\text{span}(D(x))$, the prediction on the SSD subspace $\text{span}(\tilde{D}(x))$ matches the behavior of the system exactly. This is a direct consequence of the fact that $\text{span}(\tilde{D}(x))$ 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 4.2, EDMD with dictionary D also predicts the functions in $\text{span}(\tilde{D})$ exactly. However, its prediction for functions outside of $\text{span}(\tilde{D})$ leads to large errors.

Example 8.2: (Duffing Equation): Here, we investigate the efficacy of the proposed methods in approximating Koopman eigenfunctions and invariant subspaces. Consider the Duffing equation [35, Section 4.2]

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= -0.5x_2 + x_1 - x_1^3,\end{aligned} \tag{59}$$

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 (59) with

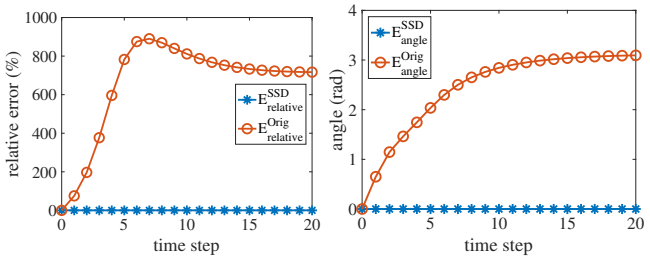

Fig. 1: Relative (left) and angle (right) prediction errors on the original and SSD subspaces for system (57) on a trajectory of length $M = 20$.

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 $\phi(x) \equiv 1$. 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 7.3. 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 7.3. 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 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 2(left) predicts that the trajectories eventually converge to one of the stable equilibria. Figure 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.

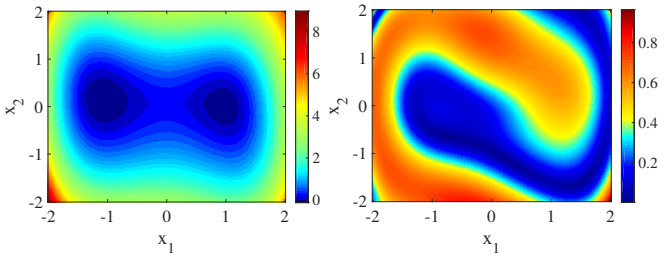

Fig. 2: The approximated eigenfunction corresponding to the eigenvalue $\lambda = 0.9919$ (left) and the absolute value of the approximated eigenfunctions corresponding to the eigenvalues $\lambda = 0.9989 \pm 0.0037j$ (right) for the Koopman operator associated with (59), as identified by the Approximated-SSD algorithm.

We use the relative and angle errors defined in (58) 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 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.

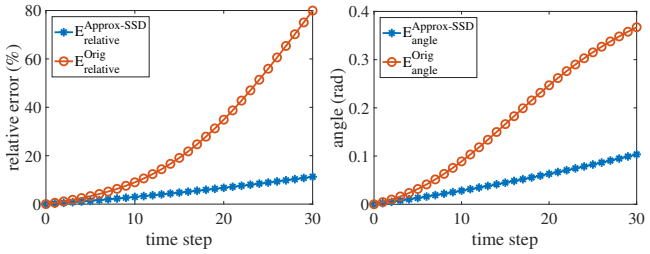

Fig. 3: Relative (left) and angle (right) prediction errors on Approximated-SSD and original subspaces for system (59) on a trajectory of length $M = 30$.

IX. Conclusions

We have studied 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 EDMD 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 SSD 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 instead approximations of the Koopman eigenfunctions and invariant subspaces. Future work will develop parallel and distributed counterparts of the algorithms proposed here over network systems, obtain out-of-sample bounds on prediction accuracy for the output of the Approximated-SSD algorithm, investigate the design of noise-resilient methods to identify Koopman eigenfunctions and invariant subspaces, and explore methods to expand the original dictionary to ensure the existence of non-trivial invariant subspaces.

References

[1] 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.

[2] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[3] B. O. Koopman and J. V. Neumann, "Dynamical systems of continuous spectra," *Proceedings of the National Academy of Sciences*, vol. 18, no. 3, pp. 255–263, 1932.

[4] I. Mezić, "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dynamics*, vol. 41, no. 1-3, pp. 309–325, 2005.

[5] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, pp. 115–127, 2009.

[6] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism," *Chaos*, vol. 22, no. 4, p. 047510, 2012.

[7] A. Surana, M. O. Williams, M. Morari, and A. Banaszuk, "Koopman operator framework for constrained state estimation," in *IEEE Conf. on Decision and Control*, Melbourne, Australia, 2017, pp. 94–101.

[8] M. Netto and L. Mili, "A robust data-driven Koopman Kalman filter for power systems dynamic state estimation," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7228–7237, 2018.

[9] A. Mauroy and J. Goncalves, "Koopman-based lifting techniques for nonlinear systems identification," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2550–2565, 2020.

[10] 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*, Montreal, Canada, May 2019, pp. 6244–6250.

[11] 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*, vol. 16, no. 2, pp. 1164–1196, 2017.

[12] 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*, vol. 440, no. 2, pp. 750–772, 2016.

[13] 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*, vol. 28, no. 3, pp. 985–1010, 2018.

[14] A. Alla and J. N. Kutz, "Nonlinear model order reduction via dynamic mode decomposition," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. B778–B796, 2017.

[15] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[16] H. Arbabi, M. Korda, and I. Mezic, "A data-driven Koopman model predictive control framework for nonlinear flows," *arXiv preprint arXiv:1804.05291*, 2018.

[17] S. Peitz and S. Klus, "Koopman operator-based model reduction for switched-system control of PDEs," *Automatica*, vol. 106, pp. 184–191, 2019.

[18] B. Huang, X. Ma, and U. Vaidya, "Feedback stabilization using Koopman operator," in *IEEE Conf. on Decision and Control*, Miami Beach, FL, Dec. 2018, pp. 6434–6439.

[19] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," *arXiv preprint arXiv:1707.01146*, 2017.

[20] A. Sootla and D. Ernst, "Pulse-based control using Koopman operator under parametric uncertainty," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 791–796, 2017.

[21] A. Narasingam and J. S. Kwon, "Data-driven feedback stabilization of nonlinear systems: Koopman-based model predictive control," *arXiv preprint arXiv:2005.09741*, 2020.

[22] G. Mamakoukas, M. Castano, X. Tan, and T. Murphey, "Local Koopman operators for data-driven control of robotic systems," in *Robotics: Science and Systems*, Freiburg, Germany, June 2019.

[23] M. L. Castaño, A. Hess, G. Mamakoukas, T. Gao, T. Murphey, and X. Tan, "Control-oriented modeling of soft robotic swimmer with Koopman operators," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1679–1685.

[24] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[25] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[26] 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*, vol. 22, no. 6, pp. 887–915, 2012.

[27] 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*, vol. 1, no. 2, pp. 391–421, 2014.

[28] M. S. Hemati, M. O. Williams, and C. W. Rowley, "Dynamic mode decomposition for large and streaming datasets," *Physics of Fluids*, vol. 26, no. 11, p. 111701, 2014.

[29] 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*, vol. 18, no. 3, pp. 1586–1609, 2019.

[30] 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*, vol. 380, pp. 355–377, 2019.
[31] 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*, vol. 57, no. 3, p. 42, 2016.
[32] 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*, vol. 31, no. 4, pp. 349–368, 2017.
[33] M. R. Jovanović, P. J. Schmid, and J. W. Nichols, "Sparsity-promoting dynamic mode decomposition," *Physics of Fluids*, vol. 26, no. 2, p. 024103, 2014.
[34] S. L. Clainche and J. M. Vega, "Higher-order dynamic mode decomposition," *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 2, pp. 882–925, 2017.
[35] 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*, vol. 25, no. 6, pp. 1307–1346, 2015.
[36] M. Korda and I. Mezić, "On convergence of extended dynamic mode decomposition to the Koopman operator," *Journal of Nonlinear Science*, vol. 28, no. 2, pp. 687–710, 2018.
[37] 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.
[38] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox, "Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems," *Physica D: Nonlinear Phenomena*, vol. 406, p. 132401, 2020.
[39] 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*, vol. 27, no. 10, p. 103111, 2017.
[40] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," in *Conference on Neural Information Processing Systems*, 2017, pp. 1130–1140.
[41] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *American Control Conference*, Philadelphia, PA, July 2019, pp. 4832–4839.
[42] S. E. Otto and C. W. Rowley, "Linearly recurrent autoencoder networks for learning dynamics," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, pp. 558–593, 2019.
[43] 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*, vol. 11, no. 2, pp. 1–19, 2016.
[44] M. Korda and I. Mezic, "Optimal construction of Koopman eigenfunctions for prediction and control," *IEEE Transactions on Automatic Control*, 2020, to appear.
[45] 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," *arXiv preprint arXiv:1909.10638*, 2019.
[46] M. Haseli and J. Cortés, "Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric Subspace Decomposition," *https://arxiv.org/abs/1909.01419*, 2020.
[47] G. B. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed. New York: Wiley, 1999.
[48] M. Korda and I. Mezić, "Optimal construction of Koopman eigenfunctions for prediction and control," 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02278835
[49] X. Li, S. Wang, and Y. Cai, "Tutorial: Complexity analysis of singular value decomposition and its variants," *arXiv preprint arXiv:1906.12085*, 2019.
[50] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The Quarterly Journal of Mathematics*, vol. 11, no. 1, pp. 50–59, 1960.
[51] I. Markovsky and S. V. Huffel, "Overview of total least-squares methods," *Signal processing*, vol. 87, no. 10, pp. 2283–2302, 2007.
[52] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.
[53] R. M. Jungers and P. Tabuada, "Non-local linearization of nonlinear differential equations via polyflows," in *American Control Conference*, Philadelphia, PA, 2019, pp. 1906–1911.

APPENDIX

Here we gather some linear algebraic results. The proofs are presented in the online version [46].

Lemma A.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,

(a) $\mathcal{R}(AZ^A) = \mathcal{R}(A) \cap \mathcal{R}(B)$;
(b) Z^A and Z^B have full column rank.

Lemma A.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 $\mathcal{R}(C) \subseteq \mathcal{R}(D)$.

Lemma A.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

$$\mathcal{R}\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right) = \mathcal{R}\left(\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\right), \quad \mathcal{R}(A_1C) = \mathcal{R}(B_1C),$$

then

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

Lemma A.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 $\mathcal{R}(A\hat{C}) = \bigcap_{i=1}^{\infty} \mathcal{R}(AC_i)$.

Masih Haseli was born in Kermanshah, Iran in 1991. He received the B.Sc. degree, in 2013, and M.Sc. degree, in 2015, both in Electrical Engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. In 2017, he joined the University of California, San Diego to pursue the Ph.D. degree in Mechanical and Aerospace Engineering. His research interests include system identification, nonlinear systems, network systems, data-driven modeling and control, and distributed and parallel computing. Mr. Haseli was the recipient of the bronze medal in Iran's national mathematics competition in 2014.

Jorge Cortés (M'02, SM'06, F'14) received the Licenciatura degree in mathematics from Universidad de Zaragoza, Zaragoza, Spain, in 1997, and the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Madrid, Spain, in 2001. He held postdoctoral positions with the University of Twente, Twente, The Netherlands, and the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He was an Assistant Professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, from 2004 to 2007. He is currently a Professor in the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA. He is the author of Geometric, Control and Numerical Aspects of Nonholonomic Systems (Springer-Verlag, 2002) and co-author (together with F. Bullo and S. Martínez) of Distributed Control of Robotic Networks (Princeton University Press, 2009). He is a Fellow of IEEE and SIAM. His current research interests include distributed control and optimization, network science, resource-aware control, nonsmooth analysis, reasoning and decision making under uncertainty, network neuroscience, and multi-agent coordination in robotic, power, and transportation networks.