Fast Identification of Koopman-Invariant Subspaces: Parallel Symmetric Subspace Decomposition

Masih Haseli and Jorge Cortés

Abstract— **This paper presents a parallel data-driven method to identify finite-dimensional subspaces that are invariant under the Koopman operator describing a dynamical system. Our approach builds on Symmetric Subspace Decomposition (SSD), which is a centralized scheme to find Koopman-invariant subspaces and Koopman eigenfunctions. Given a dictionary of functions, a collection of processors communicating through a strongly connected time-invariant directed graph, and a set of data snapshots gathered from the dynamical system, our approach distributes the data snapshots among the processors and initializes each processor with the original dictionary. Then, at each iteration, processors prune their dictionary by using the information received from their neighbors and applying the SSD method on the pruned dictionary with their local data. We prove that the algorithm terminates in a finite number of iterations and that the processors, upon termination, reach consensus on the maximal Koopman-invariant subspace in the span of the dictionary (and is therefore equivalent to SSD). A simulation example shows significant gains in time complexity by the proposed method over SSD.**

I. Introduction

Advances in data storage, processing, acquisition, and analytics have driven a surge of activity in data-driven learning and modeling of dynamical phenomena. Neural networks and state-space models are two mainstream approaches to model dynamical systems. With enough data, neural networks can describe the dynamics accurately; however, they are not convenient tools to analyze them analytically. On the other hand, the state-space approach can provide mathematical models useful for analysis. Such models however are generally nonlinear and the difficulty of their analysis grows drastically as the dimension of the state space increases. The Koopman operator is an alternative approach to simplify the analysis of dynamical systems. However, its infinite-dimensional nature prohibits the use of existing efficient numerical methods developed to work with digital computers. One can resolve this issue by working in finite-dimensional subspaces that are invariant under the application of the Koopman operator. Identifying such subspaces for real-time applications is the problem considered here.

Literature Review: The Koopman operator [1], [2] is a linear but generally infinite-dimensional operator that fully characterizes the behavior of a dynamical system. As a consequence of its linearity and considering the fact that its eigenfunctions evolve linearly in time, the Koopman operator is a powerful tool for analyzing nonlinear dynamical systems [3], [4]. This leads to a wide range of applications, including system identification [5], state estimation [6], and control of nonlinear dynamical systems [7]–[10]. Despite these appealing applications, the infinite-dimensional nature of the Koopman operator has prevented its widespread use due to the lack of computational methods to identify and represent it. In recent years, there have been several breakthroughs to circumvent this issue. Those approaches generally form two main groups. The first group is comprised of methods that approximate the action of the operator on a finite-dimensional subspace such as Dynamic Mode Decomposition (DMD) [11] and Extended Dynamic Mode Decomposition (EDMD) [12]. DMD is a method that can identify temporal evolutions using sequential data gathered from a linear time-invariant dynamics [11]. The works [13]–[15] generalize DMD to work with non-sequential and noisy data. EDMD is an extension of DMD that can approximate the projection of the action of the Koopman operator on a finite-dimensional subspace spanned by a predefined dictionary of functions [12]. The work [16] studies the convergence of EDMD to the Koopman operator as the dimension of the predefined subspace and the number of available data snapshots go to infinity. EDMD has been extended to work with noisy data [17] and to reduce its computational complexity using kernel methods [18]. DMD and EDMD are able to approximate linear evolutions in the dynamics but they are not useful for long term predictions due to the error in the approximations. As a result, it is imperative to develop methods that can identify subspaces that are invariant under the Koopman operator, which is the subject of the second group of approaches. The works [19], [20] provide empirical methods to approximate Koopman eigenfunctions which span Koopman-invariant subspaces. The works [21], [22] present methods based on neural networks to perform this task. However, it is important to note that these methods do not provide analytical guarantees for those subspaces to be Koopman invariant. Our recent works [23], [24] provides necessary and sufficient conditions for the identification of linear evolutions according to the dynamics based on the application of EDMD forward and backward in time. These conditions led us to propose the Symmetric Subspace Decomposition (SSD) strategy. Under generic assumptions, this algorithm provably finds the maximal Koopman-invariant subspace contained in the span of a given dictionary. The work [24] also presents the Streaming Symmetric Subspace Decomposition (SSSD) algorithm, an equivalent online method to SSD that enables to work with large and streaming data sets.

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

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

Statement of Contributions: We present a parallel data-driven method to identify the Koopman-invariant subspaces associated with a potentially nonlinear discrete-time dynamics. Moreover, the proposed algorithm is able to identify the functions that evolve linearly in time (also known as Koopman eigenfunctions) according to the dynamics. The presented approach is parallel in nature and can be implemented on parallel processing hardware, ensuring fast computations for real-time applications. The starting point of our iterative strategy are a dictionary of functions, a collection of processors communicating through a strongly connected time-invariant directed graph, and a set comprised of data snapshots gathered from the dynamical system. The proposed procedure introduces a rule to distribute the available data snapshots among the processors and uploads the original dictionary to every processor. At each iteration, each processor receives the dictionary of its neighbors and calculates the intersection of the subspaces spanned by those dictionaries and its own dictionary. Then the processor applies SSD on that subspace using its local data snapshots and prunes its dictionary using the basis of the Koopman-invariant subspace. We study the evolution of the processors' dictionaries to characterize the algorithm properties. We prove that the algorithm terminates in a finite number of iterations, where all processors reach consensus on the largest Koopman-invariant subspace in the span of the original dictionary. This establishes the equivalence of the proposed strategy with the centralized SSD strategy. Finally, we illustrate the superior performance of our method versus SSD using a simulation example of a nonlinear polynomial vector field. All proofs are omitted for space reasons and will appear elsewhere[1].

II. Preliminaries

In this section, we gather basic definitions on the Koopman operator and graph theory.

Koopman Operator Theory: Our exposition here follows [4]. Let $T : \mathcal{M} \to \mathcal{M}$ be a time-invariant mapping defined over $\mathcal{M} \subseteq \mathbb{R}^n$. Consider the discrete-time dynamics

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

Note that the dynamics acts on the points of $\mathcal{M}$, generating trajectories. Instead, the Koopman operator is an alternative description of the dynamics that acts on functions (also known as observables). Formally, let $\mathcal{F}$ be a linear space of functions defined on $\mathcal{M}$ and taking values in $\mathbb{C}$. Let $\mathcal{F}$ be 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

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

The linearity of $\mathcal{F}$ results in the linearity of $\mathcal{K}$, i.e., for every $f_1, f_2 \in \mathcal{F}$ and $c_1, c_2 \in \mathbb{C}$, we have

$$\mathcal{K}(c_1 f_1 + c_2 f_2) = c_1 \mathcal{K}(f_1) + c_2 \mathcal{K}(f_2). \tag{3}$$

Despite its linearity, the Koopman operator completely captures the global characteristics of (1) if $\mathcal{F}$ contains the functions describing the states of the system. Formally, given $f_i(x) := x_i$, we must have $f_i \in \mathcal{F}$ for every $i \in \{1, \dots, n\}$. This condition together with (2) may force $\mathcal{F}$ to be infinite dimensional.

The function $\phi \in \mathcal{F}$ is an *eigenfunction* of $\mathcal{K}$ with *eigenvalue* λ if

$$\mathcal{K}(\phi) = \lambda \phi. \tag{4}$$

A notable property of the Koopman eigenfunctions is their linear evolution in time, i.e., $\phi(x^+) = (\phi \circ T)(x) = \mathcal{K}(\phi)(x) = \lambda \phi(x)$. The linear evolution of Koopman eigenfunctions in conjunction with (3) simplifies the analysis of the nonlinear system (1), since one can use the spectral properties of the Koopman operator. Formally, given a set of eigenfunctions $\{\phi_i\}_{i=1}^{N_k}$ with corresponding eigenvalues $\{\lambda_i\}_{i=1}^{N_k}$, consider a function f in $\operatorname{span}(\{\phi_i\}_{i=1}^{N_k})$, i.e.,

$$f = \sum_{i=1}^{N_k} c_i \phi_i, \tag{5}$$

for some $\{c_i\}_{i=1}^{N_k} \subset \mathbb{C}$. Then,

$$f(x(k)) = \sum_{i=1}^{N_k} c_i \lambda_i^k \, \phi_i(x(0)), \, \forall k \in \mathbb{N}. \tag{6}$$

The elements $\{c_i\}_{i=1}^{N_k}$ are called *Koopman modes* associated with f and $\{\phi_i\}_{i=1}^{N_k}$. Since the Koopman operator is generally infinite dimensional, one might need to use $N_k = \infty$ in order to fully describe the action of the operator.

The subspace $\mathcal{S} \subseteq \mathcal{F}$ is *Koopman-invariant* under the application of the Koopman operator if for every $f \in \mathcal{S}$, we have $\mathcal{K}(f) \in \mathcal{S}$. Moreover, $\mathcal{S}$ is the *maximal Koopman-invariant* subspace of $\mathcal{L} \subseteq \mathcal{F}$ if it contains all the invariant subspaces in $\mathcal{L}$. Trivially, a set of eigenfunctions spans a Koopman-invariant subspace.

Graph Theory: We follow the exposition in [25, Chapter 1]. The pair $G = (V, E)$ is a *directed graph* of order m, where V is a set comprised of m *nodes* and $E \subseteq V \times V$ is a set comprised of ordered pairs of nodes called *edges*. Given an edge from node i to node j, $(i, j) \in E$, we say that i is an *in-neighbor* of j and j is an *out-neighbor* of i. A *path* in the graph G with *length* p is a sequence of $p+1$ nodes such that the ordered pair comprised of every two consecutive nodes

[1]Throughout the paper, we use the following notation. We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}$, and $\mathbb{C}$, the sets of natural, nonnegative integer, real, and complex numbers respectively. For a matrix $A \in \mathbb{C}^{m \times n}$, we denote the sets comprised of its rows, the set comprised of its columns, the number of its rows, and the number of its columns by $\operatorname{rows}(A)$, $\operatorname{cols}(A)$, $\sharp\operatorname{rows}(A)$, and $\sharp\operatorname{cols}(A)$ respectively. We denote its transpose and range space by A^T and $\mathcal{R}(A)$ respectively. Moreover, if $n = m$ we use A^{-1} to denote the inverse of A. For $a_1, \dots, a_n \in \mathbb{C}$, we denote by $\operatorname{diag}(a_1, \dots, a_n)$, the diagonal matrix with $a_1, \dots, a_n$ on the main diagonal. Given matrices $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{m \times d}$, we denote by $[A, B] \in \mathbb{C}^{m \times (n+d)}$ the matrix created by concatenating A and B. Given $v_1, \dots, v_k \in \mathbb{C}^n$, $\operatorname{span}\{v_1, \dots, v_k\}$ represents the set comprised of all vectors in the form of $c_1 v_1 + \cdots + c_n v_n$, with $c_1, \dots, c_n \in \mathbb{C}$. Given sets A and B, $A \subseteq B$ means that A is a subset of B. Moreover, we denote by $A \cap B$ and $A \cup B$, the union and intersection of A and B. Given functions $f : B \to A$ and $g : C \to B$, $f \circ g : C \to A$ denotes their composition. Given integers a, b we denote by $a \bmod b$, the remainder of division of a by b. Given a directed graph, we denote by $\mathcal{N}_{\text{in}}(\text{i})$, the set comprised of the in-neighbors of node i.

is an edge of the graph. We say a path is *closed* if it starts and finishes with the same node. A node in a direct graph is called *globally reachable* if there exists a path from every other node to it. A directed graph is *strongly connected* if every node is globally reachable.

III. Problem Statement

Our objective is to develop fast data-driven methods to identify finite-dimensional Koopman invariant subspaces associated with a dynamical system. To achieve this goal, we seek to take advantage of parallel computation to ensure a high computational efficiency and compatibility with embedded systems hardware such as graphics processing units. In this section we describe in detail the elements of the problem setup leading to the problem statement.

We start by specifying the data collected from the unknown dynamical system (1). Consider N data snapshots

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

are available. These data form matrices $X, Y \in \mathbb{R}^{N \times n}$, where x_i^T and y_i^T are ith rows of X and Y, respectively. Consistent with the Koopman approach, the second ingredient of the problem setup is a dictionary $D : \mathcal{M} \to \mathbb{R}^{1 \times N_d}$,

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

of N_d functions $d_1, \dots, d_{N_d}$ defined from $\mathcal{M}$ to $\mathbb{R}$. Note that every dictionary $\tilde{D}$ whose functions belong to $\operatorname{span}\{d_1, \dots, d_{N_d}\}$ can be completely characterized by a matrix C with N_d rows and D as $\tilde{D}(x) = D(x)C$. The effect of dictionary D on data matrices is

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

Throughout the paper, we consider the following assumption.

Assumption III.1: (Full Column Rank Dictionary Matrices): The matrices $D(X)$ and $D(Y)$ have full column rank. □

This assumption requires the functions of the dictionary to be linearly independent. Also, it requires the data snapshots to be diverse enough to encapsulate the behavior of the dynamics.

The final ingredient of the problem setup is a group of M processors (agents) communicating according to a strongly connected time-invariant directed graph G. The dictionary snapshots are distributed among the agents

$$D(X_i), D(Y_i),\ i \in \{1, \dots, M\},$$

such that

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

Moreover, the agents share full rank signature data matrices $D(X_s)$ and $D(Y_s)$ such that for every $i \in \{1, \dots, M\}$

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

Such partitioning can be done in a number of alternative ways (e.g., by uploading the signature data snapshots to agents and evenly dividing the rest of the data among the processors). We are finally ready to formulate the problem statement.

Problem Statement: Given the dictionary $D = [d_1, \dots, d_{N_d}]$, the data snapshots $X, Y \in \mathbb{R}^{N \times n}$, and the group of M processors, we seek to design a provably correct parallel algorithm to ensure that the processors find a dictionary $\tilde{D} \subset \operatorname{span}\{d_1, \dots, d_{N_d}\}$ spanning the largest Koopman-invariant subspace contained in $\operatorname{span}\{d_1, \dots, d_{N_d}\}$. The processors can only use local data and rely on communication with neighbors to achieve this objective.

Throughout the paper, we rely on data-driven methods that are not specifically designed to work with noisy data. Consequently, one might need to preprocess the data before using the proposed algorithms.

IV. Background: Symmetric Subspace Decomposition

Here, we review [23], [24] a centralized algorithmic solution to identify the maximal Koopman-invariant subspace in the span of a dictionary. The algorithm, termed Symmetric Subspace Decomposition (SSD), will be useful in devising a solution for the problem stated in Section III.

Algorithm 1 Symmetric Subspace Decomposition [23, Algorithm 1]

Initialization
$i \leftarrow 1$, $A_1 \leftarrow D(X)$, $B_1 \leftarrow D(Y)$, $C_{\mathrm{SSD}} \leftarrow I_{N_d}$
while 1 **do**
  $\begin{bmatrix} Z_i^A \\ Z_i^B \end{bmatrix} \leftarrow \operatorname{null}([A_i, B_i])$ ▷ Basis for the null space
  if $\operatorname{null}([A_i, B_i]) = \emptyset$ **then**
    return 0 ▷ The basis does not exist
    break
  end if
  $n_i^A \leftarrow$ number of rows of Z_i^A
  $m_i^A \leftarrow$ number of columns of Z_i^A
  if $n_i^A \leq m_i^A$ **then**
    return C_{SSD} ▷ The procedure is complete
    break
  end if
  $C_{\mathrm{SSD}} \leftarrow C_{\mathrm{SSD}} Z_i^A$ ▷ Reducing the subspace
  $A_{i+1} \leftarrow A_i Z_i^A$, $B_{i+1} \leftarrow B_i Z_i^A$, $i \leftarrow i + 1$
end while

For convenience, we define the action of the SSD algorithm on matrices $D(X), D(Y)$ by

$$C_{\mathrm{SSD}} = \mathrm{SSD}(D(X), D(Y)). \tag{7}$$

The output of the SSD algorithm has the following property.

Theorem IV.1: (Symmetric Subspace Decomposition [23, Theorem V.4]): Given Assumption III.1, the SSD algorithm has the following properties:

(a) The matrix C_{SSD} is either 0 or has full column rank and satisfies $\mathcal{R}(D(X)C_{\mathrm{SSD}}) = \mathcal{R}(D(Y)C_{\mathrm{SSD}})$;

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

If $C_{\rm SSD} \neq 0$, one can define a new dictionary as

$$\tilde{D}(x) = D(x)C_{\rm SSD}, \quad \forall x \in \mathcal{M}. \tag{8}$$

According to Theorem IV.1(a) and Assumption III.1, there exists a nonsingular square matrix K that satisfies

$$\tilde{D}(Y) = \tilde{D}(X)K. \tag{9}$$

The eigendecomposition of K fully characterizes the functions in the span of $\tilde{D}(x)$ that evolve linearly in time, i.e., given $\lambda \in \mathbb{C}$ and $w \in \mathbb{C}^{\tilde{N}_d} \setminus \{0\}$ with $Kw = \lambda w$, we have

$$\tilde{D}(Y)w = \lambda \tilde{D}(X)w. \tag{10}$$

Hence, the function $\phi(x) := \tilde{D}(x)w$ satisfies $\phi(T(x)) = \lambda\phi(x)$ for every $x \in \text{rows}(X)$. It is important to note that, $\phi(x)$ is not necessarily an eigenfunction of the Koopman operator since we only know that it evolves linearly on $\text{rows}(X)$, not $\mathcal{M}$ (we refer the reader to [24] regarding necessary and almost sure sufficient conditions for ϕ to be a Koopman eigenfunction).

The next result shows that (10) and the eigendecomposition of K can also fully characterize the linear evolutions in the original dictionary matrices $D(X)$ and $D(Y)$.

Theorem IV.2: (Identification of Linear Evolutions using the SSD Algorithm [23, Theorem V.5]): Under Assumption III.1, let $\tilde{D} : \mathcal{M} \to \mathbb{R}^{1\times\tilde{N}_d}$ be the dictionary defined by $\tilde{D}(x) = D(x)C_{\rm SSD}$, $x \in \mathcal{M}$. Then $\tilde{D}(Y)w = \lambda\tilde{D}(X)w$ for some $\lambda \in \mathbb{C}$ and $w \in \mathbb{C}^{\tilde{N}_d}$ if and only if there exists $v \in \mathbb{C}^{N_d}$ such that $D(Y)v = \lambda D(X)v$. In addition $v = C_{\rm SSD}w$.

V. Parallel Symmetric Subspace Decomposition

The SSD algorithm does not solve the problem stated in Section III because it is centralized. Here, we propose the Parallel Symmetric Subspace Decomposition (P-SSD) strategy which can be executed over a group of processors that use local data and communication. We analyze the properties of the P-SSD algorithm, showing that it achieves the same solution as SSD while running more efficiently.

A. The P-SSD algorithm

We provide pseudocode for the steps of the P-SSD algorithm in Algorithm 2. Here we describe the various steps. Given M processors communicating according to a time-invariant strongly connected direct graph and given the dictionary snapshots $D(X), D(Y)$, the P-SSD algorithm first forms the signature snapshot matrices $D(X_s), D(Y_s)$ with full column rank from $D(X), D(Y)$ and uploads them to every processor (one always can find a set of signature data snapshots with the size at most $2N_d$ based on Assumption III.1). Then, the rest of the available data snapshots are distributed to the processors. This can be done in a number of ways taking into account the impact on the algorithm execution. For instance, in the case of homogeneous processors, one could distribute the data as evenly as possible to minimize the maximum number of data snapshots available to processors and, consequently, the time that takes for agents to complete one iteration of their procedure. After distributing the data snapshots among the processors, each agent runs the procedure in Steps 4-18 of Algorithm 2. In that procedure, each agent receives the dictionary of its neighbors, and uses the SSD algorithm to find the largest Koopman invariant subspace in the intersection of the subspaces spanned by its dictionary and its neighbor's dictionaries. Finally, the agent updates its dictionary by a basis for the computed subspace and transmit that basis to its neighbors. The function $\text{basis}(\mathcal{S})$ returns a full column rank matrix whose columns provide a basis for $\mathcal{S}$. If $\mathcal{S}$ only contains the zero vector, then $\text{basis}(\mathcal{S})$ returns 0 (we define $\sharp\text{cols}(0) := 0$).

Algorithm 2 Parallel Symmetric Subspace Decomposition

Initialization

1: Create full column rank signature snapshots: $D(X_s), D(Y_s)$
2: Distribute the rest of data among agents and form: $D(X_i), D(Y_i), \forall i \in \{1, \ldots, M\}$
3: Upload the signature snapshots to the agents:
$D(X_i) \leftarrow [D(X_s)^T, D(X_i)^T]^T$
$D(Y_i) \leftarrow [D(Y_s)^T, D(Y_i)^T]^T$

End Initialization

Procedure for agent i:

4: $k \leftarrow 0$, $C_0^i \leftarrow I_{N_d}$, $\text{flag}_0^i \leftarrow 0$
5: **while** 1 **do**
6: $k \leftarrow k + 1$
7: $\text{flag}_k^i \leftarrow 0$
8: Receive C_{k-1}^j, $\forall j \in \mathcal{N}_{\rm in}(\text{i})$
9: $D_k^i \leftarrow \text{basis}\Big(\bigcap_{j\in\{i\}\cup\mathcal{N}_{\rm in}(\text{i})} \mathcal{R}(C_{k-1}^j)\Big)$
10: $E_k^i \leftarrow \text{SSD}(D(X_i)D_k^i, D(Y_i)D_k^i)$
11: **if** $\sharp\text{cols}(D_k^iE_k^i) < \sharp\text{cols}(C_{k-1}^i)$ **then**
12: $C_k^i \leftarrow D_k^iE_k^i$ ▷ The subspace gets smaller.
13: **else**
14: $C_k^i \leftarrow C_{k-1}^i$ ▷ The subspace does not change.
15: $\text{flag}_k^i \leftarrow 1$
16: **end if**
17: Transmit C_k^i to out-neighbors
18: **end while**

B. Convergence of P-SSD and Equivalence with SSD

Here, we characterize the properties of the P-SSD algorithm. Since the algorithm runs in parallel, it is imperative to introduce a way to detect if the algorithm has reached an equilibrium, meaning that the agents do not change their outputs anymore. The next result shows that the flag variables take care of this task.

Proposition V.1: (Equilibrium and Flags): In the P-SSD algorithm, $\text{flag}_l^i = 1$ for some $l \in \mathbb{N}$ and for every $i \in \{1, \ldots, M\}$ if and only if $C_k^i = C_l^i$ and $\text{flag}_k^i = 1$ for every $i \in \{1, \ldots, M\}$ and every $k \geq l$.

The next result shows the monotonicity of the matrix of each agent with respect to the matrices maintained by its in-neighbors in the previous iteration.

Lemma V.2: (Subspace Monotonicity of Agents' Matrices): In the P-SSD algorithm, for each iteration $k \in \mathbb{N}$, every $i \in \{1, \dots, M\}$ and every $j \in \{i\} \cup \mathcal{N}_{\text{in}}(\text{i})$, we have $\mathcal{R}(C_k^i) \subseteq \mathcal{R}(C_{k-1}^j)$.

The following result shows that the algorithm reaches an equilibrium in a finite number of iterations where all agents agree on the same range space.

Theorem V.3: (Reaching Consensus Equilibrium after finite iterations): For a strongly connected graph and under Assumption III.1, the P-SSD algorithm reaches a consensus equilibrium after a finite number of iterations, i.e., there exists $l \in \mathbb{N}$ such that, for all $k \geq l$, $\text{flag}_k^i = 1, \forall i \in \{1, \dots, M\}$ and $\mathcal{R}(C_k^1) = \mathcal{R}(C_k^2) = \cdots = \mathcal{R}(C_k^M)$.

Theorem V.3 in combination with Proposition V.1 show that the agents reach an equilibrium consensus after finite iterations, i.e., the range space, on which the agents agree on, stays the same along time.

Next, we show that the range space on which agents agree under P-SSD is the same as the one found by SSD, thereby establishing the equivalence between both algorithms.

Theorem V.4: (Equivalence of P-SSD and SSD): Let $C_{\text{SSD}} = \text{SSD}(D(X), D(Y))$. For a strongly connected graph and under Assumption III.1, let l be the iteration at which the P-SSD algorithm reaches consensus. Then, for all $k \geq l$ and all $i \in \{1, \dots, M\}$, we have $\mathcal{R}(C_k^i) = \mathcal{R}(C_{\text{SSD}})$ and $\mathcal{R}(D(X)C_k^i) = \mathcal{R}(D(Y)C_k^i)$.

After reaching the consensus at iteration l, one can use the matrices $C_l^i, i \in \{1, \dots, M\}$ instead of C_{SSD} in (8) to form the new dictionary that satisfies (9) for some K. The eigendecomposition of K fully characterizes the linear evolutions in $D(X)$ and $D(Y)$ according to Theorem IV.2.

VI. Simulation Results

Here, we illustrate the effectiveness of the P-SSD algorithm and compare it against the SSD algorithm. Let

$$x_1^+ = 0.8\, x_1 \tag{11a}$$
$$x_2^+ = 0.5\, x_2 + 0.7\, x_1^2 + 0.1, \tag{11b}$$

with $x = [x_1, x_2]^T$. This system is actually a discrete-time Polyflow [26] and has a Koopman-invariant subspace comprised of polynomial functions. We use the dictionary $D(x) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]$ with $N_d = 10$. We use $N = 5 \times 10^5$ data points randomly selected from $[-2, 2] \times [-2, 2]$. We use the first 10 data snapshots in our data set to form the signature dictionary snapshots.

To identify the maximal Koopman-invariant subspace in the span of D associated with system (11), we use the SSD and P-SSD strategies with $M \in \{5, 10, 100\}$ agents communicating according to a directed ring graph. The implementation of P-SSD is done on a single computer in MATLAB®. We calculate the time taken to complete an iteration as the maximum time taken among all agents to complete the iteration. Moreover, we use the `tic` and `toc` commands to calculate the elapsed time for each agent.

For all $M \in \{5, 10, 100\}$ the P-SSD algorithm reaches the consensus equilibrium after two iterations. Moreover, both SSD and P-SSD methods find the maximal 6-dimensional Koopman-invariant subspace spanned by $\{1, x_1, x_2, x_1^2, x_1x_2, x_1^3\}$. Using the output matrix of any agent instead of C_{SSD} in (8), one can define the new dictionary $\tilde{D}(x)$. Now, by finding the matrix K using (9) and calculating its eigendecomposition, one can find the eigenfunctions associated with system (11) and verify analytically that they are evolving linearly in time. Table I shows the eigenfunctions and their corresponding eigenvalues. Since $x_1, x_2 \in \text{span}(\tilde{D}(x))$, the eigenfunctions completely describe the evolution of system (11) in a linear manner, i.e., one can use (6) to predict the behavior of functions in form of (5) or one simply can create the linear dynamical system

$$\boldsymbol{\phi}(x^+) = \Lambda \boldsymbol{\phi}(x), \forall x \in \mathbb{R}^2,$$

with $\boldsymbol{\phi}(x) = [\phi_1(x), \dots, \phi_6(x)]^T$ and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_6)$ to describe the behavior of (11).

TABLE I: Identified eigenfunctions and eigenvalues of the Koopman operator associated with system (11).

Eigenfunction	**Eigenvalue**
$\phi_1(x) = 1$	$\lambda_1 = 1$
$\phi_2(x) = x_1$	$\lambda_2 = 0.8$
$\phi_3(x) = x_1^2$	$\lambda_3 = 0.64$
$\phi_4(x) = 25\, x_1^2 - 5\, x_2 + 1$	$\lambda_4 = 0.5$
$\phi_5(x) = x_1^3$	$\lambda_5 = 0.512$
$\phi_6(x) = 25\, x_1^3 - 5\, x_1x_2 + x_1$	$\lambda_6 = 0.4$

Figure 1 illustrates the time taken by the P-SSD and SSD strategies. The P-SSD algorithm for $M = 5$, $M = 10$, and $M = 100$, is 78%, 90%, and 98% faster, respectively, than the SSD algorithm.

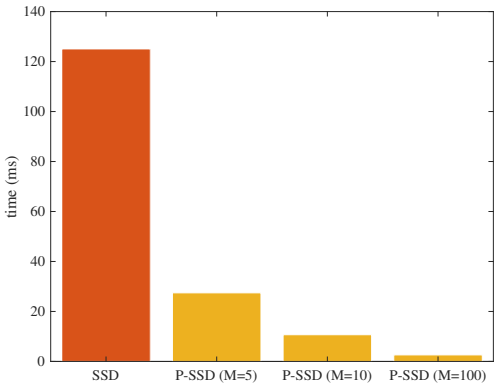

Fig. 1: Elapsed time to identify the maximal Koopman-invariant subspace in the span of the dictionary D for system (11).

To show the superiority of our method to the existing algorithms, we compare the prediction error of P-SSD with the prediction error associated with EDMD [12] on the original dictionary. The EDMD prediction matrix is defined as $K_{\text{EDMD}} = \text{argmin}_K \|D(Y) - D(X)K\|_F^2 = D(X)^\dagger D(Y)$.

Given the initial state x_0, we define the following relative and angle prediction error at time step $k \in \mathbb{N}_0$ to compare the accuracy P-SSD and EDMD in long term prediction.

$$E^{\text{EDMD}}_{\text{relative}}(k) = \frac{\left\| D(x(k)) - D(x(0))\left(K_{\text{EDMD}}\right)^k \right\|_2}{\|D(x(k))\|_2} \times 100,$$

$$E^{\text{P-SSD}}_{\text{relative}}(k) = \frac{\left\| \tilde{D}(x(k)) - \tilde{D}(x(0))\left(K\right)^k \right\|_2}{\|\tilde{D}(x(k))\|_2} \times 100,$$

$$E^{\text{EDMD}}_{\text{angle}}(k) = \angle\Big(D(x(k)), D(x(0))\left(K_{\text{EDMD}}\right)^k \Big),$$

$$E^{\text{P-SSD}}_{\text{angle}}(k) = \angle\Big(\tilde{D}(x(k)), \tilde{D}(x(0))\left(K\right)^k \Big).$$

Figure 2 shows the effectiveness of P-SSD compared to EDMD in accurate multi-step prediction of the trajectories starting from a random initial condition. This superiority is a direct consequence of P-SSD identifying and operating on a Koopman-invariant subspace.

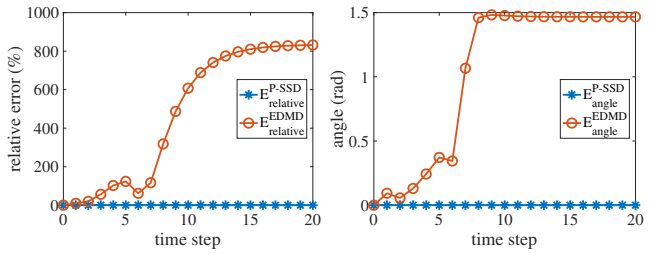

Fig. 2: Relative (left) and angle (right) prediction errors for EDMD and SSD for system (11) on a trajectory of length $L = 20$.

VII. Conclusions

We have presented a parallel algorithm to identify from data, the functions that evolve linearly in time according to unknown nonlinear dynamics. We have shown that the proposed strategy, termed P-SSD, is equivalent to the SSD algorithm, which is a centralized method that identifies the maximal Koopman-invariant subspace in the span of a predefined dictionary. However, the P-SSD strategy can be implemented on parallel computing hardware and, as we have observed in simulations, is much faster than SSD. For future work, we plan on characterizing analytically the time and computational complexity of the proposed algorithm and on extending it to work with streaming data sets. In addition, we plan on analyzing its robustness against addition and deletion of nodes in the graph and packet losses due to imperfect communication between the processors.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] A. Mauroy and J. Goncalves, "Linear identification of nonlinear systems: A lifting technique based on the Koopman operator," in *IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec. 2016, pp. 6500–6505.

[6] 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.

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

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

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, "A kernel-based method for data-driven Koopman spectral analysis," *Journal of Computational Dynamics*, vol. 2, no. 2, pp. 247–265, 2015.

[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] 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.

[21] 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.

[22] 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.

[23] 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.

[24] ——, "Learning Koopman eigenfunctions and invariant subspaces from data: symmetric subspace decomposition," *IEEE Transactions on Automatic Control*, 2020, submitted. Available at https://arxiv.org/abs/1909.01419.

[25] F. Bullo, J. Cortés, and S. Martinez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009.

[26] 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.