Fast Online Reinforcement Learning Control using State-Space Dimensionality Reduction

Tomonori Sadamoto\textsuperscript{1}, Member, IEEE, Aranya Chakrabortty\textsuperscript{2}, Senior Member, IEEE, and Jun-ichi Imura\textsuperscript{3}, Senior Member, IEEE

Abstract—Reinforcement Learning (RL) is an effective way of designing model-free linear quadratic regulator (LQR) controller for linear time-invariant (LTI) networks with unknown state-space models. However, when the network size is large, conventional RL can result in unacceptably long learning time. In this paper we resolve this problem by developing an alternative approach for RL-based LQR that combines dimensionality reduction with RL theory. The approach is to construct a compressed state vector by projecting the measured state through a projective matrix. This matrix is constructed from the state measurements, and can be viewed as an empirical controllability gramian that captures the level of redundancy in the controllability of the open-loop network model. Next, a RL-controller is learned using the reduced-dimensional state instead of the original state such that the resultant cost is close to the optimal LQR cost. Numerical benefits as well as the cyber-physical implementation benefits of the proposed approach are verified using illustrative examples including an example of wide-area control of the IEEE benchmark power system.

Index Terms—Large-scale Networks, Reinforcement Learning, Dimensionality Reduction

I. INTRODUCTION

Learning theory has been pursued in the domain of control systems since the early 1970’s, mostly through system identification and adaptive control \cite{1, 2, 3, 4}. Starting from the 90’s control theorists started taking an active interest in relating learning with optimal control \cite{5}. In particular, reinforcement learning (RL), which was originally developed in the artificial intelligence community, was related to optimal control as they both seek to find policies that minimize cost functions for a given set of tasks \cite{6}. In recent years, RL has been refurbished with renewed enthusiasm in the context of linear quadratic regulator (LQR) designs with model uncertainties through several papers such as \cite{7, 8, 9, 10}, and surveys such as \cite{11, 12}.

The fundamental idea behind RL-based LQR is to iteratively learn the optimal state-feedback control gain matrix directly from online measurements of the states and inputs without knowing the system model. The system is initially excited with exploratory noise, followed by gathering of sufficient number of state and input samples for solving a least-squares estimate of the Riccati gain matrix. This estimate is thereafter iterated upon by updating a nonlinear RL function till the estimate converges to produce the optimal control gain. Curse of dimensionality, however, continues to be an ongoing debate for all RL-based control designs. For example, the least squares step requires at least \(n(n+1)/2\) number of data samples for a unique solution, \(n\) being the order of the system, which can take an unacceptably long amount of learning time if \(n\) is large. A few approaches have been proposed to overcome this curse. For example, in \cite{9, 13} the authors have proposed a method to learn decentralized RL-controllers by regarding the interference between various subsystems in a network as an uncertainty. Each decentralized controller is learned independently, and closed-loop system stability is guaranteed using the small-gain theorem. This method, however, is mostly applicable for weakly-connected networks. It does not apply to networks where subsystems share a strong coupling as commonly seen in many real-world applications such as electric power systems, transportation systems, multi-agent coordination control of swarms, financial networks, and so on. In the artificial intelligence community, on the other hand, a notion called state aggregation has been proposed to overcome the computational bottleneck of RL in controlling Markov decision processes (MDP). The idea here is that instead of the original state-space, the decision-maker finds decision vectors in an abstract state-space much faster by treating groups of states as a unit, ignoring irrelevant state information. A number of abstractions have been proposed, for example see \cite{14, 15, 16, 17}, with a brief survey in \cite{18}. However, these abstraction methods are not based on any control-theoretic property of a network, nor do they answer quantitative questions such as “what system-theoretic information is lost when an abstraction is applied?”, or “how do the dynamics of the network model decide the level of abstraction”, and so on.

In this paper we propose a new approach for fast design of RL controllers using the concept of dimensionality reduction. The basic idea is to exploit low-rank property of the controllability gramian of the system, project the measured states into a corresponding lower-dimensional space that captures the dominant eigenvectors of the gramian, and learn a LQR controller in this lower-dimensional space. Low-rank property of the controllability gramian in this case means that following a disturbance, one would need to control only the dominantly controllable behavior of the network states, and still be able to steer the network to its desired mission. Practically speaking,
all large-scale spatially distributed networks in reality exhibit such low-rank controllability property to one extent or another. The most pertinent example, again, is an electric power system, where this low-rank property manifests in the form of low-frequency inter-area oscillations in the power outputs of synchronous generators, characterizing the dominant behavior of the grid after major faults. Based on this observation, our proposed approach is to construct a matrix from online measurements of states and inputs that projects these measurements to a low-dimensional data space, capturing the dominant traits of the network dynamics. The matrix construction is done using singular value decomposition of the measurements. A RL controller is then designed based on this compressed data. Because of its lower dimensionality, the computational complexity for learning is drastically reduced. Stability and optimality of the control performance are theoretically evaluated using robust control theory by treating the dimensionality-reduction error as an uncertainty. The effectiveness of the proposed learning method is illustrated by numerical simulations of a group of agents trying to achieve model-free formation control, and also by model-free power oscillation damping control of the IEEE benchmark power system.

Note that a dimensionality-reduction based RL controller was recently proposed in [19] under the assumption that the network model exhibits a singular perturbation structure. In contrast, the design in this paper does not require any such assumption on the network. Also, model-based control designs using dimensionality reduction have been proposed in earlier papers such as [20] as well as in more recent papers such as [21]. However, their extension to a completely model-free execution such as ours has not been reported yet.

The rest of the paper is organized as follows. Section II formulates the problem of fast RL control design, followed by Section III which presents an algorithm for this design assuming that the dimensionality reduction is exact. Theoretical results on closed-loop stability and performance degradation depending on the reduction error are derived in Section IV. An extension of the proposed algorithm to networks with semi-stable dynamics is presented in Section V, followed by numerical simulations in Section VI. Section VII concludes the paper.

**Notation:** We denote the set of real numbers as $\mathbb{R}$, the pseudo-inverse of a full-row rank matrix $P$ as $P^+$, the range space spanned by the column vectors of a matrix $P$ as $\text{im}(P)$, the null space spanned by those as $\ker(P)$, and the $n$-dimensional identity matrix as $I_n$. The subscript $n$ is omitted if obvious. We denote the $i$-th column vector of $I_m$ by $e_i^m$, and $n$-dimensional row vector whose all entries are one by $1_n$. Given $P \in \mathbb{R}^{n \times n}$ where $n \leq n$, the matrix $P$ is defined such that $P^+P + P^T P = I$. When $n = n$, $P$ is empty. The operator $\otimes$ denotes the Kronecker product. For a matrix $P := [p_1, \ldots, p_n]$, $\text{vec}(P) := [p_1^T, \ldots, p_n^T]^T$. The Dirac-delta function is denoted as $\delta(t)$. The $L_2$-norm of a square integrable function $v(t) \in \mathbb{R}^n$ is defined by $\|v(t)\|_{L_2} := \left(\int_0^\infty v^T(t)v(t)dt\right)^{1/2}$. The $H_\infty$-norm of a stable proper transfer matrix $G$ is defined by $\|G(s)\|_{H_\infty} := \sup_{s \in \mathbb{R}} \|G(j\omega)\|$ where $\| \cdot \|$ denotes the induced 2-norm, and the $H_2$-norm of a stable strictly proper transfer matrix $G$ is defined by $\|G(s)\|_{H_2} := \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \text{tr}(G(j\omega)G^T(-j\omega))d\omega\right)^{1/2}$.

**A. Brief Review of Off-Policy Iteration**

Consider a linear networked dynamical system consisting of $L$ subsystems. For $l \in \{1, \ldots, L\}$, the $l$-th subsystem dynamics is described as

$$\Sigma_l : \dot{x}_{l[t]} = A_{ll}x_{l[t]} + \sum_{j \in \mathcal{N}_l} A_{lj}x_{j[t]} + B_1u_{l[t]}, \quad x_{l[0]}(0) = x_{l[0]}$$

(1) where $x_{l[t]} \in \mathbb{R}^{n_l}$ is a state, $u_{l[t]} \in \mathbb{R}^{m_l}$ is a control input, $\mathcal{N}_l$ is the index set of subsystems connecting to the $l$-th subsystem, and $x_{l[0]}$ is an initial state of the subsystem. Using the notation

$$A := \begin{bmatrix} A_{11} & \cdots & A_{1L} \\ \vdots & \ddots & \vdots \\ A_{L1} & \cdots & A_{LL} \end{bmatrix}, \quad B := \begin{bmatrix} B_1 \\ \vdots \\ B_L \end{bmatrix}$$

$$n := \sum_{l=1}^L n_l, \quad m := \sum_{l=1}^L m_l, \quad x := [x_{1[T]}^T, \ldots, x_{L[T]}^T]^T, \quad u := [u_{1[T]}^T, \ldots, u_{L[T]}^T]^T,$$

the interconnected network model can be written as

$$\Sigma : \dot{x} = Ax + Bu, \quad x(0) = x_0.$$ (3)

We impose the following three assumptions on (3).

**Assumption 1:** The matrices $A$ and $B$ are unknown.

**Assumption 2:** The state $x$ is measurable.

**Assumption 3:** The open-loop system in (3) is stable, i.e., $A$ is Hurwitz.

Assumption 1 implies that although we know that the system of our interest is LTI we do not know its model. Assumptions 1-2 imply that both $n$ and $m$ are known. Our goal is to design a state-feedback controller

$$u = -Fx$$ (4)

such that the cost function

$$J := \int_0^\infty x^T(t)Qx(t) + u^T(t)Ru(t)dt$$ (5)

where $Q \succeq 0$ and $R > 0$ are chosen matrices, is minimized. If $A$ and $B$ were known one can find the optimal controller $F$ in (4) by solving an algebraic Riccati equation. However, since both of these matrices are completely unknown, the standard LQR approach no longer applies. Instead a reinforcement learning (RL) based approach needs to be used, as shown in recent papers such as [9], [22].

Several variants of RL exist in the literature. One popular method is the so-called off-policy iteration [9], which consists of two stages, namely - *Online Data Collection* and *Policy Improvement*. In the first stage, one excites the system with an exploration noise, say denoted as $u(t) = \sin(t^2)$ for $0 \leq t \leq 1$, $u(t) = 0$ for $t > 1$, and collects the time-series measurements of $x(t)$ driven by this noise and the initial state disturbance $x_0$. The purpose of adding the noise is to persistently excite
x(t). For details of this step please see the page 14 in [9]. In the second stage, one estimates the control gain matrix F iteratively using the measured data from stage 1. This stage essentially solves the algebraic Riccati equation associated with (5) using Newton’s method based on x(t) and u(t) [22]. Note that for this estimation only the dataset before starting the policy improvement is needed. In other words, one does not need to collect x(t) any more once the controller is updated. Once the controller has converged, we apply the resultant controller to the system. The complete off-policy iteration algorithm is summarized for the sake completeness and later reference in Algorithm 1 as follows.

Algorithm 1: Off-policy iteration [9]

**Initialization:**
Given N, j ← 0, k ← 0, F₀ = 0, ε ≥ 0, fix a set {t_j}j∈{0,...,N−1} such that t_{N−1} > ... > t₀ ≥ 0.

**Online Data Collection:**
For j = 0,...,N−1, do:
1. Measure x(t) for t ∈ [t_j, t_{j+1}] under a given u(t).
2. Numerically compute
   \[ φ_j := (x(t_{j+1}) ⊗ x(t_{j+1}) − x(t_j) ⊗ x(t_j))^T ∈ \mathbb{R}^{1×n^2} \]  
   \[ ρ_j := \int_{t_j}^{t_{j+1}} (x(t) ⊗ u(t))^T dt ∈ \mathbb{R}^{1×nm} \]  
   \[ σ_j := \int_{t_j}^{t_{j+1}} (x(t) ⊗ x(t))^T dt ∈ \mathbb{R}^{1×n^2} \]

**Policy Improvement:**
3. Find W_k and F_{k+1} satisfying
   \[ Θ_k \begin{bmatrix} vec(W_k) \\ vec(F_{k+1}) \end{bmatrix} = z_k \]  
   with
   \[ Θ_k := [\phi - 2ρ(I ⊗ R) − 2σ(I ⊗ F_k^T R)] ∈ \mathbb{R}^{N×(2n^2 + nm)} \]
   \[ z_k := −σvec(Q + F_k^T RF_k) ∈ \mathbb{R}^{N} \]
   where φ ∈ \mathbb{R}^{N×n^2}, ρ ∈ \mathbb{R}^{N×nm}, and σ ∈ \mathbb{R}^{N×n^2} are the stacked versions of φ_j, ρ_j, and σ_j, respectively.
4. Exit if ||F_{k+1} − F_k|| < ϵ, otherwise return to 3.

The following theorem from [9] shows that the off-policy iteration in Algorithm 1 produces an optimal controller.

**Theorem 1:** [9] Consider Σ in [4] and J in [5]. Assume
   \[ \text{rank}[ρ σ] = \frac{n(n + 1)}{2} + mn. \]  
   where ρ and σ are the stacked versions of ρ_j in [7] and σ_j in [5]. Then, the following two statements hold:
   i) A − BF_k is Hurwitz for any integer k > 0, where F_k follows from the solution of (7).
   ii) Define \( F_∞ := \lim_{k→∞} F_k \). The control u = −F_∞ x minimizes J.

The main limitation of Algorithm 1 is that it is not easily scalable to high-dimensional networks with a large value of n. Step 3 of the algorithm is the main computational bottleneck. To solve (9), one needs to compute the pseudo-inverse of Θ_k in (10). The computational cost for that based on singular value decomposition is in order of \( O(\min\{N^2n^4, n^2N^2\}) \) [23], where N is the number of collected data samples. To satisfy (12), one must also have N ≥ \( \frac{n(n+1)}{2} + mn \), which makes the overall cost of the algorithm to be \( O(n^6) \). Moreover, the pseudo-inverse must be computed at every iteration of the policy improvement. Thus, the learning time, defined as the time needed for running the Policy Improvement of Algorithm 1, may be unacceptably large, thereby limiting its application for real-time control.

B. Purpose and Approach of This Paper

Motivated by this problem, in this paper we propose an alternative approach for designing the model-free controller [4] such that it makes the cost function J in [5] as small as possible while reducing the learning time to significantly lesser than \( O(n^6) \). Our fundamental approach is to utilize the low-rank property of the controllability gramian of the open-loop network. In many large-scale networked systems, disturbances and control inputs tend to excite only a part or a combination of the state variables [24], implying that the collected data x(t) associated with a low-dimensional controllable subspace is sufficient for learning. As long as the data is compressed appropriately from the viewpoint of controllability, learning based on the low-dimensional data can be expected to work well. Our goal is to theoretically establish this intuition while at the same time developing an algorithm to learn F using low-dimensional data. The proposed approach is briefly summarized as follows:

1) First, we construct a reduced-dimensional compressed state vector ξ as
   \[ ξ := Px, \]
   by projecting the measured state trajectory x(t) through a full row-rank matrix P ∈ \( \mathbb{R}^{n×n} \) where \( n ≪ n \). The matrix P will capture the level of redundancy in the controllability of the network model [3] that allows for dimensionality reduction. Note that the system model is not known, so P will be constructed solely from the measurements of x(t).

2) Next, a controller [4] is learned using the reduced-dimensional state ξ(t) instead of the original state x(t) such that the performance cost of the resulting closed-loop system is as close to the optimal cost J in [5] as possible.

In the following sections, we show how these steps are achieved.

III. PROPOSED ALGORITHM

A. A Sufficient Condition for Lossless Compression

We first consider an ideal situation when the data compression is lossless, i.e. x(t) can be recovered from the compressed data ξ(t). In that case, one can expect that the learning based on the lossless data ξ(t) will provide an optimal controller minimizing J in [5]. This is indeed true, as shown next. A
sufficient condition for the lossless compression is to reduce the redundant states of $x$ in terms of controllability. This is summarized as the following lemma.

**Lemma 1**: Consider $\Sigma$ in (3). Let $P$ satisfy
\begin{equation}
\text{im} P^T = \text{im} \mathcal{R}(A, [B \ 0])
\end{equation}
where $\mathcal{R}(A, b) := [b, Ab, \ldots, A^{n-1}b]$. Then, $\xi$ in (13) satisfies
\begin{equation}
\bar{\Sigma} : \bar{\xi} := PAP^T \xi + PBu, \quad \xi(0) = Px_0,
\end{equation}
where $PAP^T$ is Hurwitz. Furthermore,
\begin{equation}
x(t) = P^T\xi(t), \quad \forall t
\end{equation}
holds for any $u(t)$ and $x_0$.

**Proof**: See [25].

This lemma implies that if $P$ satisfies (14), the $\hat{n}$-dimensional state $\xi$ exactly captures the behavior of the $n$-dimensional state $x$. From (16), it follows that the cost function with respect to the compressed state $\xi$
\begin{equation}
\bar{J} := \int_0^\infty \hat{\xi}^T(t)\hat{Q}\hat{\xi}(t) + u^T(t)Ru(t)dt, \quad \hat{Q} := (P^T)^TQP^T
\end{equation}
is identical to $J$ in (5). In other words, the learning for $n$-dimensional system $\Sigma$ in (3) with a cost function $J$ by using $\xi$ is equivalent to the learning for the reduced-order system (15) with a cost function $\bar{J}$ in (17). This can be done by employing the off-policy iteration method of Algorithm 1 if a matrix $P$ satisfying (14) is found.

**B. Construction of $P$ from Data**

We next show an approach by which one can construct $P$ satisfying (14) by using the measured data $x(t)$. We first introduce the following lemma.

**Lemma 2**: Consider $\Sigma$ in (3). If $P$ satisfies (14), the condition
\begin{equation}
(I - P^T P)x(t) \equiv 0, \quad \forall t
\end{equation}
holds for any $u$ and $x_0$. In addition, without loss of generality, we suppose that $\hat{n}$ is the minimal dimension for satisfying (18). Then, (18) is equivalent to (14).

**Proof**: First, we show the sufficiency. If (14) holds, by multiplying $P^T P$ to (16) from the left, we have $P^T P x(t) \equiv 0$ for any $u$, $x_0$, and $t$. This is equivalent to (18). Next, we show (14) if (18) holds. To this end, we first show
\begin{equation}
P^T P \mathcal{R}(A, [B \ 0]) = \mathcal{R}(A, [B \ 0]).
\end{equation}
Since (18) follows for $u = 0$, we have
\begin{equation}
P^T P e^{At} x_0 = e^{At} x_0, \quad \forall t.
\end{equation}
Substituting $t = 0$ in (20), we have $P^T P x_0 = x_0$. Next, taking one time derivative of (20) and substituting $t = 0$, we have $P^T P A x_0 = A x_0$. Repeating this procedure up to the $(n-1)$-th order time-derivative, we have $P^T P A^i x_0 = A^i x_0$ for $i \in \{0, \ldots, n - 1\}$. Next, consider an impulse input applied to the $i$-th control port, described as
\begin{equation}
u(t) = e^{|_{i}} \delta(t).
\end{equation}
Since (18) also holds for this input, we have
\begin{equation}
P^T P e^{At} (x_0 + B_i) = e^{At} (x_0 + B_i), \quad \forall t,
\end{equation}
where $B_i$ denotes the $i$-th column of $B$. By substituting $t = 0$, we have $P^T P B_i = B_i$, where the relation $P^T P x_0 = x_0$ is used. Repeating the same procedure for all $i \in \{1, \ldots, n\}$ and up to the $(n-1)$-th order time-derivative, the relation (19) follows. Clearly, (19) is equivalent to $\text{im} P^T \subseteq \text{im} \mathcal{R}(A, [B \ 0])$. Thus, if $\hat{n}$ is minimal for satisfying (18), this becomes (14). This completes the proof.

Lemma 2 implies that the $P$ satisfying (18) spans a reachable subspace from $u$ and $x_0$. However, finding a $P$ which satisfies (18) exactly for any arbitrary input signal $u$ can be difficult. Therefore, we consider solving the following online minimization problem
\begin{equation}
\min_{\mathcal{P} \in \mathbb{R}^{n \times n}} \sum_{j=0}^{N-1} \| (I - P^T P)x(j) \|_T^2,
\end{equation}
where $\hat{n} \leq n$ is a given design parameter, and $x(j)$ is the sampled state driven by initial state disturbance $x_0$ and an exploratory noise $u(t)$. The $x(t)$ measurements that are collected during the Online Data Collection step of Algorithm 1 can be used in (22) for finding $P$. One can see that the solution of (22) approximately satisfies (18) if $N$ is sufficiently large. This minimization problem can be solved by using the singular value decomposition of
\begin{equation}
X := [x(t_0), \ldots, x(t_{N-1})] \in \mathbb{R}^{n \times N},
\end{equation}
with $P^T$ given by the first $\hat{n}$ columns of the left-singular matrix.

Another option can be to construct $P$ offline, i.e., before the disturbance hits the network. That can further save online computation time. Offline construction of $P$, for example, can be done by using the empirical controllability gramian [26]. For $i \in \{1, \ldots, m\}$, let $x_i(t)$ be the response of the system $\Sigma$ with the impulse input given by (21). The empirical controllability gramian is defined as
\begin{equation}
\Phi_{\text{emp}}(t) := \sum_{i=1}^{m} \int_0^t (x_i(\tau) - \bar{x}_i(\tau))^T (x_i(\tau) - \bar{x}_i(\tau)) d\tau
\end{equation}
with $\bar{x}_i(t) := \frac{1}{T} \int_0^t x_i(\tau) d\tau$. It can be clearly seen that
\begin{equation}
\lim_{t \to \infty} \Phi_{\text{emp}}(t) = \Phi(A, B)
\end{equation}
where $\Phi(\cdot, \cdot)$ is the standard controllability gramian. This implies that we can construct the standard controllability gramian by using $m$ sets of $\{x(t), u(t)\}$ without knowing the system model. Although we do not know what disturbance is injected, it may be possible to know a possible subspace $\mathcal{X} \subseteq \mathbb{R}^n$ in which $x_0$ lies. In this situation, similarly as the above, we can construct another empirical controllability gramian associated with the disturbance. Let $d_1, \cdots, d_r$ be the basis of the distribution $\mathcal{X}$, i.e., $\text{ini}[d_1, \ldots, d_r] = \mathcal{X}$. Let $x_i(t)$ be the response of $\Sigma$ with the disturbance of $x(0) = d_i$ for $i \in \{1, \ldots, r\}$. Denote the empirical controllability gramian (24) by $\Phi^{\star}_{\text{emp}}$. Similarly, denote the gramian (24) associated
with the input by \( \Phi^{u}_{\text{emp}} \). Then, by choosing eigenvectors of \( \Phi_{\text{emp}} := \Phi^{u}_{\text{emp}} + \Phi^{c}_{\text{emp}} \) corresponding to the non-zero eigenvalues, we can construct \( P \) satisfying (19) for any \( x_0 \in \mathcal{X} \).

### C. Preconditioned Off-Policy Iteration

Based on the idea of constructing the projection matrix \( P \) from \( x(t) \), and compressing \( x \) to \( \xi \) as described above, we next propose a revised version of the off-policy iteration. We refer to this method as *preconditioned* off-policy iteration. Unlike the regular off-policy iteration this method consists of three stages, namely - Online Data Collection, Preconditioning, and Policy Improvement. The first stage is identical to that in Algorithm 1. In the second stage, we define \( \hat{\phi}, \hat{\rho}, \hat{\sigma} \) by replacing \( x \) in (6)-(8) with \( \xi \). The revised matrices are computed as

\[
\hat{\phi} = \phi(P^T \otimes P^T) \in \mathbb{R}^{1 \times \hat{n}^2}
\]

\[
\hat{\rho} = \rho(P^T \otimes I) \in \mathbb{R}^{1 \times \hat{m} \hat{n}}
\]

\[
\hat{\sigma} = \sigma(P^T \otimes P^T) \in \mathbb{R}^{1 \times \hat{n}^2}
\]

(26)-(28)

The third stage is similar to the policy improvement stage of Algorithm 1, but the vector \( x \) is replaced with \( \xi \) by using the relation (16). The pseudo-code of the proposed algorithm is as follows.

**Algorithm 2:** Preconditioned off-policy iteration

**Initialization and Online Data Collection:**
Same as in Algorithm 1.

**Preconditioning:**
1. Given \( \hat{n} \), find \( P \) by SVD of (23).
2. Compute \( \hat{\phi}, \hat{\rho}, \text{and } \hat{\sigma} \) in (26)-(28).

**Policy Improvement:**
3. Find \( \hat{W}_k \) and \( \hat{F}_{k+1} \) satisfying

\[
\hat{\Theta}_k \begin{bmatrix} \text{vec}(\hat{W}_k) \\ \text{vec}(\hat{F}_{k+1}) \end{bmatrix} = \hat{z}_k
\]

(29)

with

\[
\Theta_k := \left[ \hat{\phi} - 2\hat{\rho}(I \otimes R) - 2\hat{\sigma}(I \otimes Rloomberg) \right] \in \mathbb{R}^{N \times (2\hat{n}^2 + \hat{m} \hat{n})}
\]

(30)

\[
\hat{z}_k = -\hat{\sigma} \text{vec}(\hat{Q} + \hat{F}_k \hat{R}_k) \in \mathbb{R}^N
\]

(31)

where \( \hat{\phi}, \hat{\rho}, \hat{\sigma} \) are defined as (26)-(28), and \( \hat{Q} \) is in (17).

4. Go to the next step if \( \|\hat{F}_{k+1} - F_k\| < \epsilon \), otherwise, return to 3.
5. Return

\[
F_k = \hat{F}_k P
\]

(32)

Similar to Proposition 1, the convergence of Algorithm 2 and the optimality of the final controller are guaranteed as follows.

**Theorem 2:** Consider \( \Sigma \) in (3), and a \( P \) that satisfies (18). Assume

\[
\text{rank}[\hat{\rho} \hat{\sigma}] = \frac{\hat{n}(\hat{n} + 1)}{2} + \hat{m} \hat{n},
\]

(33)

where \( \hat{\rho} \) and \( \hat{\sigma} \) are defined in (27)-(28). If Algorithm 2 is applied to \( \Sigma \), then the following two statements hold:

i) \( A - BF_k \) is Hurwitz for any integer \( k > 0 \), where \( F_k \) is given by (32).

ii) Define \( F_{\infty} := \lim_{k \to \infty} F_k \). The control \( u = -F_{\infty} \) minimizes \( J \) in (5).

**Proof:** From the proof of Theorem 2.3.12 in [9], it follows that the control \( u = -F_k \xi \) stabilizes \( \Sigma \) in (15), i.e., \( PA^{P^T} P^T A - PB \xi F_k \) is Hurwitz for any integer \( k > 0 \). Let \( \xi := \overline{P} x \). Applying the coordinate transformation from \( x \) to \( \xi \) in (34) to the dynamics (5), we obtain

\[
\begin{bmatrix} \xi \\ \zeta \end{bmatrix} = \begin{bmatrix} PA P^T & PA \overline{P} \\ 0 & \overline{P} A \overline{P} \end{bmatrix} \begin{bmatrix} \xi \\ \zeta \end{bmatrix} + \begin{bmatrix} PB \\ 0 \end{bmatrix} u,
\]

(34)

with \( \xi(0) = P x_0 \) and \( \zeta(0) = 0 \). Note that \( \overline{P} B = 0 \), \( \overline{P} x_0 = 0 \), and \( \overline{P} A \overline{P} = 0 \) follow from (14). Since \( A \) is Hurwitz, from the upper-triangular structure of (34), \( PA \overline{P} \) is also Hurwitz. Furthermore, since \( PA \overline{P} - PB F_k \) is Hurwitz, we can see that \( A - BF_k \) is Hurwitz for any \( k \). The claim i) follows from this. Furthermore, the control law \( u = -F_{\infty} \xi \) minimizes (17) subject to the dynamics (15). Finally, due to the uncontrollability of \( \xi \) as shown in (34), the control minimizes \( J \) in (5). This completes the proof.

The computational complexity of Algorithm 2 is significantly lower than that of Algorithm 1. The complexity of the preconditioning step is \( O(\min\{n^2 N, N^2 n\}) \) [23]. This may be large if \( N \) is large. However, since the left-singular eigenvectors of \( X \) are identical to those of \( X X^T \), the complexity here is reduced to only \( O(n^3) \) even if \( N > n \). Existing methods such as [22] for computing dominant left-singular vectors can further reduce this computational cost. As mentioned in Section II-A, the computational cost for solving (30) is \( O(n^6) \). Thus, as long as \( \hat{n} \) is sufficiently small, the policy improvement stage of Algorithm 2 will save a lot of learning time. This will be demonstrated through numerical simulation later.

**IV. Robustness Analysis**

In the previous section, we have assumed the matrix \( P \) to satisfy the condition (18). In reality, however, it may be very difficult to find a \( P \) which satisfies this condition exactly. Instead, one can only find a \( P \) that satisfies this condition approximately. In other words, there exists a positive number \( \epsilon \) such that

\[
\| (I - P \xi) x(t) \| \leq \epsilon
\]

(35)

for any \( t, u \) and a given \( x_0 \). The following lemma is useful for analyzing the stability and the performance of the closed-loop system with a controller that is learned under this non-ideal condition.

**Lemma 3:** Consider \( \Sigma \) in (3), \( \Sigma \) in (15) and

\[
\Delta := \begin{bmatrix} \hat{e} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} PA P^T & 0 \\ 0 & \overline{P} A \overline{P} \end{bmatrix} \begin{bmatrix} \hat{e} \\ \hat{x} \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u
\]

(36)

with \( \hat{e}(0) = 0, \hat{x}(0) = x_0 \). The state \( x \) satisfies

\[
x = P \hat{e} + \overline{P} \hat{x}, \quad \delta x := P \hat{e} + \overline{P} \hat{x}
\]

(37)
The transfer function of \( \Delta(\hat{\Sigma}) \) is defined in (17). When \( u \subseteq 0 \) the system \( \Delta \) is Hurwitz. Let \( \hat{F}_k \) be the control gain at the \( k \)-th step of Algorithm 2. If

\[
\| \hat{\Sigma}_{cl}\Delta_{ue} \|= \| \hat{\Sigma}_{cl}\Xi^TP(sI-A)^{-1}B \|_{\mathcal{H}_\infty} \leq \epsilon \| \hat{\Sigma}_{cl}\Xi \|_{\mathcal{H}_\infty} < 1.
\]

Thus, it follows from small-gain theorem that the closed-loop of \( \hat{\Sigma}_{cl} \) and \( \Delta_{ue} \) is stable.

Next, we analyze the performance achieved by the control \( u = -F_kx \). Define \( Q_\frac{1}{2} \) and \( R_\frac{1}{2} \) as the respective Cholesky factors of \( Q \) and \( R \), i.e., \( Q_\frac{1}{2}^TQ_\frac{1}{2} = Q \) and \( R_\frac{1}{2}^TR_\frac{1}{2} = R \). Define

\[
y := \begin{bmatrix} Q_\frac{1}{2}x \\ R_\frac{1}{2}u \end{bmatrix}.
\]

The closed-loop system can then be described as the interconnection of

\[
\begin{align*}
\dot{\hat{\Sigma}} = A\hat{\Sigma} + B\hat{F}_k \hat{e}, \\
\hat{F}_k = C_w + D_{\hat{e}\hat{y}}\hat{e} + D_{\hat{e}\delta}\delta, \\
\hat{y} = C_u + D_u\hat{e}.
\end{align*}
\]

and \( \Delta \) in (36), where \( \hat{\Sigma} := [\hat{e}^T, (P^T\hat{p}_x)^T, w := [u^T, \delta]^T, \)

Thus

\[
J = \| y \|_{\mathcal{L}_2} \leq \| G_{\delta y} \|_{\mathcal{H}_2} + \| G_{\delta \hat{e}} (I - \Delta G_{\hat{e}w})^{-1}\Delta G_{\delta w} \|_{\mathcal{H}_2} \]

where \( G_{\delta y} \) denotes the transfer function of (47) from the input \( \delta \) to the output \( y \), and \( \Delta \) denotes that of the system (36) with the output \( \hat{e} \) in (46). Note that \( \| G_{\delta y} \|_{\mathcal{H}_2} = J \) and

\[
\Delta G_{\delta w} = \left[ (sI - PA)^{-1} PA \right] \times \left[ sI - B P \right] (sI - A)^{-1} PB \hat{F}_k(sI - \hat{A}) \quad \text{for} \quad \left( -B \hat{F}_k(sI - \hat{A})^{-1} PB \hat{F}_k(sI - \hat{A}) \right) \|_{\mathcal{H}_2} \leq \epsilon \| \hat{\Sigma}_{cl}\Xi \|_{\mathcal{H}_\infty} < 1.
\]

The condition (45) implies that

\[
\| \hat{\Sigma}_{cl}\Delta_{ue} \|_{\mathcal{H}_\infty} \leq \epsilon, \quad \| \hat{\Sigma}_{cl}\Xi^TP(sI-A)^{-1}B \|_{\mathcal{H}_\infty} \leq \epsilon.
\]
Thus, from (50) and (51), (43) follows where
\[ \gamma := \| P \|_{\infty} (1 + \| F_k(sI - \hat{A}_F)^{-1} P x_0 \|_{\infty}). \] (52)
This completes the proof. \(\square\)

This theorem implies the following two results: first, if the compression error satisfies (40) and \( P A P^T \) is Hurwitz (despite \( A \) being unknown to the designer), the learned controller at the \( k \)-th step of Policy Improvement in Algorithm 2 guarantees the stability of the closed-loop system. Furthermore, smaller this error closer is the closed-loop performance cost to the ideal cost \( J \).

V. EXTENSION TO SEMI-STABLE SYSTEMS

We next extend Algorithm 2 to semi-stable systems. For simplicity of analysis, we assume that \( A \) in system (3), although unknown, has one semi-stable eigenvalue. This is often the case when the network exhibits a consensus property over a connected graph \([28]\). The proposed approach can be easily generalized to systems with multiple semi-stable poles. We denote the eigenvector associated with the semi-stable pole as \( v \). The main assumption behind the proposed method is that \( v \) must be known. This assumption is actually true for many real-world applications. For consensus networks, \( v \) is the vector of all ones. For power system networks, where the semi-stable eigenvector is denoted by \( v \), the semi-stable eigenvector is dictated by the goal of controlling relative states to dynamics. Moreover in many of these networks the control objective is driven by the goal of controlling relative states to dynamics. Furthermore, smaller this error closer is the closed-loop performance cost to the ideal cost \( J \).

\[ \ker Q \supseteq \text{im } v. \] (53)

Starting from these observations, we state the following proposition:

**Proposition 1:** Consider a semi-stable system \( \Sigma \) in (3), whose semi-stable eigenvector is denoted by \( v \). Construct \( P_c \in \mathbb{R}^{n \times (n-1)} \) such that
\[ (I - P_c^T P_c) v^Tx(t) = 0, \quad \forall t \] (54)
for any \( u \) and a fixed \( x_0 \). Define
\[ P = P_c v^T. \] (55)

Suppose that (33) is satisfied. If \( Q \) in (5) satisfies (53), then by applying Algorithm 2 to the semi-stable system \( \Sigma \) the following two statements hold:

i) \( A - BF_k \) is semi-stable whose semi-stable eigenvector is
\[ v, \]

ii) The control \( u = -F_{\infty} x \) minimizes \( J \) in (5).

**Proof :** Define \( \eta := v^T x \) and \( \bar{\eta} := \eta v^T x \). Applying a coordinate transformation from \( x \) to \( [\eta^T, \bar{\eta}^T]^T \), we have
\[ \begin{bmatrix} \bar{\eta} \\ \eta \\ \bar{\eta} \\ \eta \end{bmatrix} = \begin{bmatrix} v^T A v & 0 \\ v^T A v & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \bar{\eta} \end{bmatrix} + \begin{bmatrix} v^T B \\ v^T B \end{bmatrix} u, \] (56)

where the relation \( A v = 0 \) is used. Note that \( v^T A v \) is Hurwitz. Similar to Lemma 1, it follows that
\[ \text{im } P^T = \text{im } \mathcal{R}(v^T A v, v^T [B \ x_0]). \]

Therefore, applying coordinate transformation from \( \bar{\eta} \) to \( [\eta^T, \bar{\eta}^T]^T \) where \( \bar{\xi} := \bar{\eta} \mathcal{R}(v^T A v, v^T [B \ x_0]). \)

Similar to the proof of Theorem 1, it is proven that \( P A P^T - P B F_k \) is Hurwitz for any \( k \). Note that the eigenvalues of \( A - BF_k \) with \( F_k \) in (32) are identical to those of \( P A P^T - P B F_k \), plus a zero eigenvalue. Thus, \( A - BF_k \) is semi-stable. Also note that
\[ (A - BF_k)v = -BF_k P_c v^T x = 0. \]

Thus, \( u \) is the eigenvector corresponding to the semi-stable pole of \( A - BF_k \). Therefore, claim i) follows. Claim ii) follows by using the same argument as in the proof of Theorem 1. This completes the proof. \(\square\)

This proposition implies that as long as \( P_c \) satisfies (54), i.e. the data compression is lossless, Algorithm 2 with \( P \) defined in (55) can provide an optimal controller for semi-stable systems. Such a matrix \( P_c \) can be found by the singular value decomposition of \( X_c = v^T X \)

where \( X \) is defined in (28). Furthermore, even for the non-ideal case when \( \| (I - P_c^T P_c) v^T x(t) \| \leq \epsilon \), Theorem 3 with \( P \) defined in (55) still holds for semi-stable systems.

VI. NUMERICAL SIMULATION

A. Case 1: Consensus Network System

We investigate the effectiveness of the proposed algorithm through an example of a consensus network system as shown
Fig. 3. The variation of the change of $\dot{F}$

![Graph showing the variation of the change of $\dot{F}$]

Fig. 4. Computational time needed for controller design

![Graph showing computational time needed for controller design]

in Fig. 2 The network is composed of two subsystems (areas), each of which consists of 30 and 120 nodes, respectively. The fact that the network consists of two areas is only used for just describing the network structure and its dynamics. We will not utilize this fact for controller design. The dynamics of the $i$-th node is described as

$$\dot{x}_i = \sum_{j \in N_i} a_{i,j} (x_j - x_i) + b_i u_i,$$

(57)

where $x_i \in \mathbb{R}$ is the state, and $N_i$ is the index set of nodes connected to the $i$-th node. The inter-area graph structure is given as a Barabasi-Albert model [29], and the weights of the edges are randomly chosen from the range $(0, 0.5]$. The two areas are connected through four links between four nodes in each area, and their weights are assigned to be 0.1. Let control inputs be applied to the first two nodes of area 1 individually. Suppose $b_1 = b_2 = 1$ and $b_i = 0$ for $i \in \{3, \ldots, 150\}$. Clearly, the system (57) can be described as (3) with $n = 150$. Note that this network system is semi-stable, and the corresponding eigenvector is $I_n$.

The control objective here is to improve the speed of consensus among all nodes without knowing $a_{i,j}$ and $b_i$ for all $\{i, j\}$. To this end, we take $Q$ in (5) such that $x^T Q x = \alpha \sum_{i=1}^{n} (x_1 - x_i)^2$ with a scalar weight $\alpha$ and $R = I$. Let $\alpha = 50$. The sampling sequence $\{t_j\}$ in the Online Data Collection stage is taken as $t_j = 0.01 j$ for $j \in \{0, \ldots, 2000\}$. From $t = 0$ to $t = 1$ (sec), we inject the noise $u(t) = \beta \sum_{k=1}^{400} \sin(w_k t)$ with $w_k$ whose value is randomly chosen from the range $[-20, 20]$ for $k \in \{1, \ldots, 400\}$. Let $\beta = 0.05$.

We first find a matrix $P$ by utilizing the collected data $\{x(t_j)\}_{j \in \{0, \ldots, 2000\}}$. Since the consensus network system is semi-stable, we project the data onto the stable subspace. Let $\pi^T \in \mathbb{R}^{(n-1) \times n}$ be such that $\pi^T 1_n = 0$, we compute $\hat{\eta} = \pi^T x$. Next, similarly to (22), for a given $\hat{n}$ we solve $\min_{P_c \in \mathbb{R}^{n \times n}} \sum_{j=0}^{n-1} \| (I - P_c R) \hat{\eta} (t_j) \|^2$. Let $\hat{n} = 11$. The matrix $P$ is defined by (58). We conclude the preconditioning step in Algorithm 2 by computing $\phi$, $\hat{\rho}$, and $\hat{\sigma}$ in (26)-(28). The state-feedback gain $F_c$ is computed by running the Policy Improvement step. Fig. 3 shows the variation of $F_c$ in (29).

We can see from this figure that the controller gain converges by using the compressed data.

Next, we investigate the computational complexity of Algorithm 2. The computing environment is as follows: MATLAB 2017b, Intel(R) Core(TM) i7-4587U 3.00GHz, RAM 16.0GB. Fig. 4 shows the computational time taken in Policy Improvement Algorithm 2, for different values of $\hat{n}$. The computational time at the initial iteration is the time used for preconditioning. The line with stars corresponds to the case where Algorithm 1 without data compression is applied, i.e., $P_c = I$ in the above design procedure. The line shows that as high as approximately 21 seconds of real-time are needed for each step of Policy Improvement. On the other hand, the blue, red, green, and yellow lines with circles, squares, triangles, and diamonds show the case for different values of $\hat{n}$. By comparing those with the purple line, we can see that the dimensionality reduction efficiently reduces the computational time at every step. For example, the total computation time without data compression is 337 seconds whereas that with reduction of $\hat{n} = 20$ is just four seconds. From this result, the data compression can drastically reduce learning time.

Fig. 5 shows the cost $J$ computed by different choice of controllers learned for several values of $\hat{n}$. From this figure, we can see that there exists a trade-off relation between the dimensionality (i.e., computational time) and closed-loop performance. For comparison, we design an optimal controller gain $F_{opt}$ by solving a Riccati equation for the original system model [4]. In Fig. 5 the dashed line shows the cost $J_{opt}$ achieved by this ideal controller. We can see that the controller in the case where $\hat{n} = 11$ is almost optimal. Furthermore, Table VI-A shows the first five dominant eigenvalues of the open-
loop system and the closed-loop system when \( F_{18} = \hat{P}_{18}P \) and \( F_{\text{opt}} \) are used, respectively. Clearly, the second dominant eigenvalue, associated with the consensus speed, is drastically improved by the proposed RL controller.

**B. Case 2: Power System Model**

Next, we investigate how the proposed reinforcement learning controller can be effectively applied for wide-area oscillation damping of an electric power system without the knowledge of its small-signal model. We consider a power system model composed of six areas, as shown in Fig. 6. The power system composed of Area 1-5 is a benchmark model so-called IEEE 68-bus test system [30], where the first and second area is a New England Test System (NETS) and a New York power system (NYPS), respectively. The sixth area is a copy of NETS. This power system consists of 25 synchronous generators and 52 loads, interconnected via 104 buses. The loads are modeled as constant impedance loads. Following [31], the dynamics of the power system can be modeled as a nonlinear differential-algebraic equation

\[
\dot{x} = f(x, V, w), \quad 0 = g(x, V) \tag{58}
\]

where \( x := [x_1^T, \ldots, x_{25}^T]^T \in \mathbb{R}^{100}, x_k \in \mathbb{R}^4 \) is the state of the \( k \)-th generator (angle, frequency deviation, internal voltage, and field voltage), \( V := [V_1, \ldots, V_{104}] \in \mathbb{C}^{104}, V_k \) is the complex-valued voltage of the \( k \)-th bus, \( w := [w_1, \ldots, w_{25}] \in \mathbb{R}^{125} \) and \( w_k \in \mathbb{R} \) is the reference signal (control input) to the \( k \)-th generator. We suppose that the state \( x \) is available by state estimators based on Phasor Measurement Units [32]. We consider a fault modeled as a six-area, as shown in Fig. 6. The power system model consists of 25 generators, 52 loads, and 104 buses.

![Fig. 6. Power system model consisting of 25 generators, 52 loads, and 104 buses.](image)

In this paper, we proposed a new RL control strategy for large-scale network systems. The proposed control strategy can avoid the curse of dimensionality by utilizing the low-rank property of the network in terms of its controllability. In the proposed strategy, (almost) uncontrollable state variables are eliminated by projecting the state data on to the controllable subspace. This drastically improves learning time, albeit at the cost of degrading the closed-loop performance. This degradation from optimal performance is quantified in terms of the data compression error, answering the two questions that were raised in the introduction. The results are validated by two practical examples, both of which show notable speed-up of the reinforcement learning compared to LQR while guaranteeing satisfactory sub-optimal performance.

Our future work will include the extension of this method to a hierarchical structure where microscopic or local controllers and macroscopic or global controllers are learned in parallel using the data compression concept. Relevance of this design for any \( i \) in the index set of the generators. Let \( u = [u_1, \ldots, u_6]^T \). The control input is determined by

\[
u = F(x - x^*) \tag{59}
\]

where \( x^* \) is the setpoint of \( x \) in (58). Note that this setpoint is a priori known by measuring the steady state without exploiting the detail knowledge about the system model. The controller gain \( F \) is determined by the Algorithm 2, where the state deviation from the setpoint is used instead of \( x \) in the algorithm. The design parameters are as follows: We take \( Q \) in (5) such that \( x^TQx = \alpha \sum_{i=1}^{6} \omega_i^2 \) where \( \omega_i \) is the average frequency deviation of generators inside the \( i \)-th area and \( \alpha \) is a scalar weight. Let \( \alpha = 10 \) and \( R = I \) in (5). From \( t = 0 \) to \( t = 2 \) (sec), we inject the exploration noise described in VI-A to each generator through its excitation system voltage \( w_k \). Let its amplitude be taken as \( \beta = 10^{-3} \). Let sequential data be collected for \( t \in [0, 18] \) with a sampling interval 0.01 (sec). First, we investigate the effectiveness of the existing off-policy iteration (Algorithm 1). The computational time for executing the Algorithm 1 is 140.4 (sec). We suppose that the designed controller is implemented to the system at time \( t = 18 + 140.4 = 158.4 \) (sec). Fig. 7(a) shows the trajectories of the frequency deviations of all generators after the implementation of the controller. By comparing Figs. 7(a) and (b), one can see that the damping performance over the entire time-horizon of interest is very limited due to the long learning time (140.4 seconds). To reduce the computation time, we apply the Algorithm 2 to the system with \( \hat{n} = 24 \). The learning time in that case reduces to only 6.1 seconds. Fig 7(c) shows the case where the designed controller is implemented at time \( t = 24.1 \) (sec). We can see that the damping performance of the system after the controller implementation is drastically improved by the control compared to the cases shown in Figs. 7(a)-(b). Note that the controller is linear while the power system model used for the simulations is described by the nonlinear model (58). This demonstrates also the inherent robustness property of the proposed RL controller for this example.
in light of power system coherency also needs to be analyzed in more details. We also intend to extend the method to output feedback control.

ACKNOWLEDGMENT

This research was supported by CREST, JST Grant Number JPMJCR15K1, Japan. The work of the second author was partly supported by the US National Science Foundation under grant ECCS 1711004.

REFERENCES

[1] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” IEEE Transactions on neural networks, vol. 1, no. 1, pp. 4–27, 1990.

[2] M. Kostic, I. Kanellakopoulos, and P. V. Kokotovic, Nonlinear and adaptive control design. Wiley New York, 1995.

[3] P. A. Ioannou and J. Sun, Robust adaptive control. Dover Publications, 1996.

[4] P. J. Werbos, “Neural networks for control and system identification,” in Proc. of Conference on Decision and Control, 1989, pp. 260–265.

[5] M. Vidyasagar and R. L. Karandikar, “A learning theory approach to system identification and stochastic adaptive control,” Journal of Process Control, vol. 18, no. 3-4, pp. 421–430, 2008.

[6] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction (Second Edition). The MIT Press, 2011.

[7] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” IEEE circuits and systems magazine, vol. 9, no. 3, pp. 32–50, 2009.

[8] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” IEEE Control Systems Magazine, vol. 32, no. 6, pp. 76–105, 2012.

[9] Y. Jiang and Z. Jiang, Robust adaptive dynamic programming. Wiley IEEE Press, 2017.

[10] P. Mehta and S. Meyn, “Q-learning and pontryagin’s minimum principle,” in Proc. of Conference on Decision and Control, 2009, pp. 3598–3605.

[11] P. P. Khargonekar and M. A. Dahleh, “Advancing systems and control research in the era of ML and AI,” Annual Reviews in Control, vol. 45, pp. 1–4, 2018.

[12] B. Recht, “A tour of reinforcement learning: The view from continuous control,” 2018, available at arXiv:1806.09460.

[13] Y. Jiang and Z. Jiang, “Robust adaptive dynamic programming for large-scale systems with an application to multimachine power systems,” IEEE Transactions on Circuits and Systems, vol. 59, no. 10, pp. 693–697, 2012.

[14] J. N. Tsitsiklis and B. Van Roy, “Feature-based methods for large scale dynamic programming,” Machine Learning, vol. 22, no. 1-3, pp. 59–94, 1996.

[15] S. P. Singh, T. Iaakkola, and M. I. Jordan, “Reinforcement learning with soft state aggregation,” in Advances in neural information processing systems, 1995, pp. 361–368.

[16] S. Lin and R. Wright, “Evolutionary tile coding: An automated state abstraction algorithm for reinforcement learning,” in Proc. of Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010, pp. 43–47.

[17] D. Abel, D. E. Hershkovitz, and M. L. Littman, “Near optimal behavior via approximate state abstraction,” arXiv preprint arXiv:1701.04113, 2017.

[18] L. Li, T. J. Walsh, and M. L. Littman, “Towards a unified theory of state abstraction for mdps,” in Proc. of International Symposium on Artificial Intelligence and Mathematics, 2006, pp. 531–539.

[19] S. Mukherjee, H. Bai, and A. Chakrabortty, “On model-free reinforcement learning of reduced-order optimal control for singularly perturbed systems,” in Proc. of Conference on Decision and Control, 2018, pp. 5288–5293.

[20] P. Kokotovic, H. K. Khalil, and J. O’reilly, Singular perturbation methods in control: analysis and design. Siam, 1999, vol. 25.

[21] N. Xue and A. Chakrabortty, “Optimal control of large-scale networks using clustering based projections,” 2016, available at https://arxiv.org/abs/1609.05265.

[22] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, “Adaptive optimal control for continuous-time linear systems based on policy iteration,” Automatica, vol. 45, no. 2, pp. 477–484, 2009.

[23] M. Holmes, A. Gray, and C. Isbell, “Fast SVD for large matrices,” vol. 58, 2007, pp. 249–252.

[24] A. C. Antoulas, Approximation of large-scale dynamical systems. Siam, 2005.

[25] R. E. Kalman, “Canonical structure of linear dynamical systems,” Proc. of the National Academy of Sciences, vol. 48, no. 4, pp. 596–600, 1962.

[26] S. Lall, J. E. Marsden, and S. Glavaški, “Empirical model reduction of controlled nonlinear systems,” Proc. of IFAC World Congress, vol. 32, no. 2, pp. 2598–2603, 1999.

[27] N. Lee and A. Cichocki, “Very large-scale singular value decomposition using tensor train networks,” 2014, available at arXiv:1410.6895.

[28] M. Mesbah and M. Egerstedt, Graph theoretic methods in multiagent networks. Princeton University Press, 2010.

[29] R. Albert and A. L. Barabási, “Statistical mechanics of complex networks,” Reviews of modern physics, vol. 74, no. 1, pp. 47–97, 2002.

[30] B. Pal and B. Chaudhuri, Robust control in power systems. Springer Science & Business Media, 2006.

[31] T. Sadamoto, A. Chakrabortty, T. Ishizaki, and J. Imura, “Dynamic modeling, stability, and control of power systems with distributed energy,” IEEE Control System Magazine, vol. 39, no. 2, pp. 34–65, 2019.

[32] A. K. Singh and B. C. Pal, “Decentralized dynamic state estimation in power systems using unscented transformation,” IEEE Transactions on Power Systems, vol. 29, no. 2, pp. 794–804, 2014.