Data-Driven Structured Policy Iteration for Homogeneous Distributed Systems

Siavash Alemzadeh, Shahriar Talebi, Member, IEEE, and Mehran Mesbahi, Fellow, IEEE

Abstract—Control of networked systems, comprised of interacting agents, is often achieved through modeling the underlying interactions. Constructing accurate models of such interactions—in the meantime—can become prohibitive in applications. Data-driven control methods avoid such complications by directly synthesizing a controller from the observed data. In this article, we propose an algorithm referred to as data-driven structured policy iteration (D2SPI), for synthesizing an efficient feedback mechanism that respects the sparsity pattern induced by the underlying interaction network. In particular, our algorithm uses temporary “auxiliary” communication links in order to enable the required information exchange on a (smaller) subnetwork during the “learning phase”—links that will be removed subsequently for the final distributed feedback synthesis. We then proceed to show that the learned policy results in a stabilizing structured policy for the entire network. Our analysis is then followed by showing the stability and convergence of the proposed distributed policies throughout the learning phase, exploiting a construct referred to as the “Patterned monoid.” The performance of data-driven structured policy iteration (D2SPI) is then demonstrated using representative simulation scenarios.

Index Terms—Data-driven policy iteration, networked control systems, Patterned monoids, structured control.

I. INTRODUCTION

In recent years, there has been a renewed interest in the distributed control of large-scale systems. The unprecedented interdependence and size of the data generated by such systems have necessitated a distributed approach to policy computation in order to influence or direct their behavior and performance. In these scenarios, collective actions are often synthesized via local decisions, informed by a structured information exchange mechanism. An important roadblock for centralized control design methods is, thereby, their scalability and shortcomings in utilizing the underlying structure of large-scale interconnected systems.

Structured control synthesis in the meantime is generally an NP-hard constrained optimization problem [4]. Hence, distributed control design for large-scale systems has often been pursued not necessarily to characterize optimal policies per se but to devise efficient (possibly suboptimal) control mechanisms that exploit the inherent system structure. In parallel, recent advances in measurement technologies have made available an unprecedented amount of data, motivating how online and online data processing can be leveraged for data-driven decision making on high-dimensional complex systems.

In this work, we propose the linear–quadratic regulator (LQR)-based algorithm, coined data-driven structured policy iteration (D2SPI), to iteratively learn stabilizing controllers for unknown but identical linear dynamical systems that are connected via a network induced by the coupling in their performance. The setup is a particular realization of cooperative game-theoretic decision making (see remarks under footnote 2). This class of synthesis problems is motivated by applications such as formation flight [5] and distributed camera systems [6], where the dynamics of the network nodes (agents) cannot be precisely parameterized. D2SPI is built upon a data-driven learning phase on a subgraph in a large network. This subgraph includes the agent with maximum degree in the network and requires enabling auxiliary links within this subgraph in order to iteratively learn a stabilizing structured controller (optimal for the subgraph) for the entire network (see Fig. 1). This “extension” synthesis procedure utilizes a symmetry property of the networked systems, which we refer to as the Patterned monoid (see Section II-A).

The rest of this article is organized as follows. In Section II, we introduce the problem setup and motivation behind our

\[ O(n^3) \] complexity of solving the algebraic Riccati equation [2] and scalability issues of model-predictive control [3] are among such examples.
work and provide an overview of the related literature (see Section II-B). In Section III, we present and analyze the D2SPI algorithm, followed by the theoretical analysis in Section IV. Illustrative examples are provided in Section V. Finally, Section VI concludes this article.

Notation: The operator diag(·) makes a square diagonal matrix out of the elements of its argument; vec(·), on the other hand, takes a square matrix and stacks the lower left triangular half (including the diagonal) into a single vector. We use \( N > 0 \) (\( \geq 0 \)) to denote \( N \) as a positive (semi)definite matrix. The \( i \)th eigenvalue and spectral radius of \( M \) are denoted by \( \lambda_i(M) \) and \( \rho(M) \); \( M \) is called Schur stable when \( \rho(M) < 1 \). We say that an \( n \)-dimensional linear system parameterized by the pair \( (A, B) \) is controllable if the controllability matrix \( C = [B \ AB \cdots A^{n-1} B] \) has a full rank. We denote the Kronecker product of two matrices by \( \otimes \). For a block matrix \( F \), by \([F]_{rk}\), we imply the \( r \)th row and \( k \)th column “block” component with appropriate dimensions. An (unordered) graph is characterized by \( G = (V_G, \mathcal{E}_G) \), where \( V_G \) is the set of nodes and \( \mathcal{E}_G \subseteq V_G \times V_G \) denotes the set of edges. An edge exists from node \( i \) to \( j \) if \((i, j) \in \mathcal{E}_G\); this is also specified by writing \( j \in N_i \), where \( N_i \) is the set of neighbors of node \( i \). We denote the maximum degree of \( G \) by \( d_{\text{max}}(G) \). Finally, the graph \( G \) can be represented using matrices such as the Laplacian \( L_G \) or the adjacency \( A_G \). To distinguish between system dynamics quantities related to the entire graph and a subgraph, we utilize hat and tilde notation, respectively. A semigroup is a set and a binary operator in which the multiplication operation is associative (but its elements need not have inverses). A monoid is a semigroup with an identity element. A group is a monoid, each of whose elements is invertible. We denote the set of symmetric \( n \times n \) real matrices by \( \mathcal{S}^n \), and the set invertible ones by \( G \mathbb{L}(n, \mathbb{R}) \) —which is a group under matrix multiplication also known as the general linear group.

II. PROBLEM SETUP

Consider a network of identical agents with interdependencies induced by a network-level objective. In particular, we assume that the system contains \( N \) agents forming a graph \( G = (V_G, \mathcal{E}_G) \), where each node of the graph in \( V_G \) represents a linear discrete-time system
\[
\dot{x}_{i,t} = Ax_{i,t} + Bu_{i,t}, \quad i = 1, 2, \ldots, N \tag{1}
\]
with \( x_{i,t} \in \mathbb{R}^n \) and \( u_{i,t} \in \mathbb{R}^m \) denoting the state and input of agent \( i \) at time step \( t \), respectively. The unknown system matrices \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times m} \) are assumed to form a controllable pair. The network dynamics can compactly be represented as
\[
\dot{x}_{i,t+1} = Łx_{i,t} + B\hat{u}_t \tag{2}
\]
where \( x_{i,t} \in \mathbb{R}^n \) and \( \hat{u}_t \in \mathbb{R}^m \) are comprised of the states and inputs of the entire network, \( x_{i,t} = [x_{1,t}^T \cdots x_{N,t}^T]^T \), and \( \hat{u}_t = [u_{1,t}^T \cdots u_{N,t}^T]^T \), with \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times m} \) in block diagonal forms \( \hat{A} = I_N \otimes A \) and \( \hat{B} = I_N \otimes B \). The agents’ interconnections are represented by edges \( \mathcal{E}_G \subseteq V_G \times V_G \) that can facilitate a distributed feedback design. We do not assume that \( G \) is necessarily connected; the motivation for this becomes apparent subsequently. Let \( N_i \) denote the set of neighbors of node \( i \) in \( G \) (excluding itself). Then, based on the underlying communication graph and for any choice of positive integers \( a \) and \( b \), we define a linear subspace of real-valued \( aN \times bN \) matrices as
\[
\mathcal{U}^{ab}_{a,b}(G) := \left\{ M \in \mathbb{R}^{aN \times bN} \mid [M]_{ij} = 0 \text{ if } j \notin N_i \cup \{i\} \right\}.
\]

Without having access to the system parameters \( A \) and \( B \), we are interested in designing linear feedback gains, consistent with the desired sparsity pattern induced by the network, using data generated by (2). More precisely, given an initial condition \( \hat{x}_1 \), the distributed (structured) optimal control problem assumes the form
\[
\min_{\hat{K}} \sum_{t=1}^\infty \hat{x}_{i,t}^\top \hat{Q} \hat{x}_{i,t} + \hat{u}_{i,t}^\top \hat{R} \hat{u}_{i,t}
\]
\[
s.t. \quad (2), \quad \hat{u}_{i,t} = \hat{K} \hat{x}_{i,t}, \quad \hat{K} \in \mathcal{U}^{N,N}_{a,b}(G) \tag{3}
\]
where \( \hat{K} \) stabilizes the pair \( (\hat{A}, \hat{B}) \) (i.e., \( \rho(\hat{A} + \hat{B}\hat{K}) < 1 \)), \( \hat{R} = I_N \otimes R \), and \( \hat{Q} = I_N \otimes Q_1 + L_G \otimes Q_2 \) for some given cost matrices \( Q_1 > 0, Q_2 \geq 0 \), and \( R > 0 \). Note that \( Q \in \mathcal{U}^{N,N}_{a,b}(G) \) is positive definite. Such interdependence induced through the cost has been considered in a graph-based distributed control framework; see, for instance, [7], [8], [9], and [10]. In a nutshell, the first term in \( \hat{Q} \) encodes the cost pertinent to state regulation for each agent, while the second term captures the “disagreement” cost between the neighbors.\(^2\) In this work, we propose \(^2\)One instance of such an interactive cost among agents appears in the cooperative game setup, where agent \( i \) aims to solve the minimization problem
\[
\min_{(u_{i,t})_{t=0}^\infty, (u_{j,t})_{t=0}^\infty} J_i(\hat{x}_{1}, \hat{u}_{\cdot}) = N \sum_{t=0}^\infty \left( x_{i,t}^\top Q_1 x_{i,t} + u_{i,t}^\top R u_{i,t} + \sum_{j \in N_i}^\infty (x_{j,t} - x_{i,t})^\top Q_2 (x_{j,t} - x_{i,t}) \right)
\]
a data-guided suboptimal solution for (3), not relying on the knowledge of system parameters \( \hat{A} \) and \( \hat{B} \). Instead, our approach relies on the system’s input-state time series for synthesizing distributed feedback control on \( \mathcal{G} \). A summary of challenges in analyzing this problem is listed as follows.

1) The constrained optimization problem in (3) is, in general, NP-hard [4], [12]. In particular, the problem of stabilization by decentralized static state feedback is NP-hard if one imposes a bound on the norm of the controller [12]. Even though the result is not shown for the case with no bound on the controller parameters, the general problem is commonly believed to be a computationally hard problem. Note that stabilization is a necessity for the feasibility of the optimization problem in (3) for arbitrary initial state \( x_0 \) whenever \( Q_1 > 0 \). This is simply because the cost is unbounded for an unstable policy. Nonetheless, based on the complete knowledge of system parameters, this problem has been investigated under a variety of assumptions [7], [13], [14], [15] or approached directly with the aid of projected gradient-based policy updates [16], [17].

2) In the meantime, policies obtained via data-driven approaches do not necessarily respect the hard constraints on \( \mathbf{K} \in \mathcal{U}_{m,n}^N(\mathcal{G}) \) posed in (3). In particular, we point out that “projection” onto the intersection of the constraint imposed by the network and stabilizing controllers is not straightforward due to the intricate geometry of the set of stabilizing controllers [18].

3) Another key challenge in adopting data-driven methods for the entire network is rooted in the “curse of dimensionality” inherent in the analysis of large-scale systems. In fact, even collecting data from the entire network can be prohibitive.

4) Finally, it is often impossible in applications to pause the operation of the network for data collection or decision-making purposes.\(^3\)

**A. Structures in the Problem and Our Approach**

In this work, the sparsity requirement \( \mathbf{K} \in \mathcal{U}_{m,n}^N(\mathcal{G}) \) is considered as a hard constraint for control synthesis, and as such, the corresponding optimization is challenging in general. Hence, we shift our attention from the optimal solution of (3) toward a “reasonable” suboptimal stabilizing distributed controller with a scalable computational cost. We aim to exploit the problem structure that is incurred due to the homogeneity of system dynamics and the pattern in performance index respecting the underlying graph topology—see the definition of \( \hat{Q} \) depending on the graph Laplacian \( \hat{L}_G \) in (3). In addition, as system parameters in (2) are unknown, we adapt a Q-learning procedure to our setup that provably converges to a distributed policy with suboptimality guarantees. In the absence of the sparsity constraint \( \mathbf{K} \in \mathcal{U}_{m,n}^N(\mathcal{G}) \), a learning approach for solving the optimal control problem (3) is the well-known Q-learning procedure that was first introduced by Bradtke et al. [19], [20].

Adopting this approach for (3) would require utilizing the input-state data trajectories of the entire networked systems in (2) to implement a quasi-Newton method for iteratively updating \( \mathbf{K} \) introduced by Hewer [21]. If the system is controllable, Hewer’s algorithm converges to the globally optimal solution with a quadratic rate, and so does the Bradtke’s algorithm if, in addition, the data trajectories are “informative” enough—this is usually satisfied by a sufficient condition on the input signal referred to as “persistence of excitation.” The main issue with this approach is the fact that the policy obtained in this way will not respect the hard constraint of \( \mathbf{K} \in \mathcal{U}_{m,n}^N(\mathcal{G}) \) as was posed in optimization (3). This issue is particularly critical when a projection of the iterated policy on the set \( \mathcal{U}_{m,n}^N(\mathcal{G}) \) is not practical or even fails to be stabilizing. In addition, collecting data trajectory from entire networked system in (2) can be expensive.

Inspired by the Q-learning approach, we propose a model-free structured policy iteration scheme for the synthesis problem (3) with iterative stability, convergence, and reasonable performance guarantees. While the detailed algorithm is presented subsequently in Section III, in what follows, we summarize the key steps of our design procedure for distributed data-driven policy iteration.

1) Inherent to our distributed learning algorithm is a synthesis subproblem whose dimension is related only to the maximum degree in an underlying graph rather than the dimension of the original network. In particular, we will reason that for the learning phase, our method only requires data collection from a (specific) smaller subnetwork \( \mathcal{G}_d \subseteq \mathcal{G} \) with size \( d = d_{\text{max}}(\mathcal{G}) + 1 \). This subgraph is substantially smaller than the original graph whenever \( d_{\text{max}}(\mathcal{G}) \) is significantly smaller than \( N \), reflecting the empirical feature of many real-world networks. In the meantime, our approach requires adding temporary communication links to the subgraph \( \mathcal{G}_d \) to make it a clique during the learning phase—we will discuss why this learning phase clique is required in Step 3 below—see Fig. 1. The additional links are subsequently removed for the final data-driven feedback design. In robotic applications, such a clique can be established by temporarily increasing the transmission range for neighboring nodes of the agent with maximum degree. Since longer range information transfer is generally costly, power levels for these agents are “dialed back” to their original settings following the completion of the learning phase.

2) Then, we adapt the Q-learning technique in order to learn a specific optimal policy for this subproblem using data only from the systems in the subgraph \( \mathcal{G}_d \). In particular, the proposed algorithm learns an optimal policy for the subgraph with a (off-)diagonal pattern consisting of two distinct system-level policies \( K^* \) and \( L^* \), representing the “individual” and “cooperative” components, respectively. Specifically, for any integer \( r \geq 2 \), we can define a linear subspace of \( \mathbb{R}^{r \times n} \) as

\[
L(r \times n, \mathbb{R}) := \{ \mathbf{N}_r \in \mathbb{R}^{r \times n} \mid \mathbf{N}_r = \mathbf{I}_r \otimes (A - B) + \mathbf{I}_r \mathbf{1}_T^\top \otimes B, \text{ for some } A, B \}
\]

which is also closed under matrix multiplication. Then, the policy learned for the subgraph \( \mathcal{G}_d \) takes the form \( \mathbf{K}^* = \mathbf{I}_d \otimes \)

\(^3\)For example, consider an operational large-scale group of homogeneous aerial vehicles that need to coordinate their relative states (in addition to their respective state regulation) over a network induced by their proximity [5].
\((K^* - L^*) + \mathbb{1}_{d}^{\top} \otimes L^*\), which also implies that the closed-loop system \(\tilde{A}_{Kr} := \tilde{A} + \tilde{B}K^*\) lies in \(L(d \times n, \mathbb{R})\). This is particularly useful from a design perspective as it becomes clear in the next step.

3) Next, we define the Patterned monoid as

\[\text{PM}(r \times n, \mathbb{R}) := \left\{ N_r \in L(r \times n, \mathbb{R}) \mid \text{for some } A \in G L(n, \mathbb{R}) \cap S^n, B \in S^n \right\}.\]

Note that the Patterned monoid \(\text{PM}(r \times n, \mathbb{R})\) is indeed a submonoid of \(G L(rn, \mathbb{R})\)—following by Lemma 2—i.e., it is closed under matrix multiplications and contains the identity matrix. The next observation underscores the relevance of Patterned monoids in system analysis.

**Proposition 1:** For a Schur stable matrix \(A \in L(r \times n, \mathbb{R})\) and \(0 < Q \in S^{nn}\), let \(P\) denote the unique solution to the corresponding discrete-time Lyapunov equation, i.e., \(P = A^{\top}PA + Q\). Then, \(P \in \text{PM}(r \times n, \mathbb{R})\) if and only if \(Q \in \text{PM}(r \times n, \mathbb{R})\).

The invariance of the Lyapunov equation under the action of Patterned monoid has important implications for our data-driven synthesis for large-scale networks (see Proposition 2). Another key ingredient of our approach, motivated by utilizing the Patterned monoid in the context of Q-learning, involves allowing temporary communication links on the subgraph \(G_d\) in order to make it a clique only during the learning phase—links that are subsequently removed. It then follows that the state cost parameter \(\tilde{Q}\) for the subgraph \(G_d\) lies in \(\text{PM}(d \times n, \mathbb{R})\). Then, as the obtained closed-loop system \(\tilde{A}_{Kr}\) at each iteration of the learning phase lies in \(L(d \times n, \mathbb{R})\), Proposition 1 would imply that the associated cost matrix \(\tilde{P}_{Kr}\) must lie in \(\text{PM}(d \times n, \mathbb{R})\) with two components \(P_1 \in G L(n, \mathbb{R}) \cap S^n\) and \(P_2 \in S^n\).

4) Next, an iterative Q-learning procedure is designed based on both the patterned structure of the policy with system-level components \(K\) and \(L\), and the cost matrix with \(P_1\) and \(P_2\) components. This procedure provably converges to an optimal policy for the subgraph \(G_d\) (see Theorem 2).

5) We note that terminating the operation of the entire networked system for the purpose of data collection/learning is often infeasible in real-world applications. For example, disrupting the operation of power generators and loads for improving their respective network-level performance is highly undesirable. Therefore, in addition to learning the optimal policy for the subgraph, in this work, we aim to simultaneously devise and update a policy for the rest of the network. In this direction, our algorithm iteratively learns a “stability margin” \(\tau\) according to the learned components of the policy at each iteration (see Proposition 3). This feature, together with homogeneity of the network, facilitates extending the policy synthesis procedure to a stabilizing policy for the entire network by utilizing the individual and cooperative components learned from the subgraph \(G_d\) (see Theorem 2).

6) After the Q-learning procedure has converged, the learning phase is terminated. The framework now allows removing the temporary links added to \(G_d\) during the “clique subgraph learning phase,” and the stability of the final learned policy will be guaranteed (see Corollary 1).

7) Finally, note that the learned policy for the entire graph will be, in general, a suboptimal solution to the optimization in (3). Yet, we provide guarantees on its suboptimality gap in Theorem 3 and illustrate its numerical performance in Section V.

The distributed control underpinning of the method proposed in this work follows its model-based analogues studied in [7] and [9]. In this work, our contribution is to build on these approaches and propose a model-free structured policy iteration algorithm, which is not only computationally efficient but also practical for operational large-scale networked systems.

**B. Related Literature**

Distributed control is a well-established area of research in systems theory. The roots of the field trace back to the economics literature in 1970s [22] and early works in the control literature followed suit later during that decade [23]. The main motivation for these works was lack of scalability in centralized planning and control, due to information or computational limitations [24], [25]. Fast forward a few decades, sufficient graph-theoretic conditions were provided for the stability of formations comprised of identical vehicles [26] and, along the same lines, graph-based distributed controller synthesis was further examined independently in works such as [7], [10], [16], and [27]. The topic was also studied from the perspective of spatial invariance [15], [28] and a compositional layered design [29], [30]. Moreover, from an agent-level perspective, the problem has been tackled for both homogeneous systems [7], [9], [10] and, more recently, heterogeneous ones [31].

Having access to the underlying system model is a common assumption in the literature on distributed control, where the goal is to find a distributed feedback mechanism that conforms to an underlying network topology. However, deriving dynamic models from first principles could be restrictive for large-scale systems and complex mission scenarios [32]. Such restrictions also hold for parametric perturbations that occur due to inefficient modeling or other unknown design factors. For instance, even the LQR solution with its strong input robustness properties may have small stability margins for general parameter perturbations [33]. Robust synthesis approaches could alleviate this issue when the perturbations follow specific models in both centralized [34] and distributed [35] cases. However, if the original estimates of system parameters are inaccurate or the perturbations violate the presumed model, then both stability and optimality of the proposed feedback mechanisms can be compromised. Data-driven control, on the other hand, circumvents such drawbacks and utilizes the available data generated by the system when its model is unavailable. This point of view has historically been examined in the context of adaptive control and...
system identification [36], particularly, when asymptotic properties of the synthesized system are of interest. For more recent works that have adopted a nonasymptotic outlook on data-driven control, we mention [37], [38], and [39] that used batched data for synthesis, as well as online iterative procedures [40], [41]. Furthermore, in regard to the adaptive nature of such algorithms, there is a close connection between online data-driven control and reinforcement learning [19], [42]. In these latter works, policy iteration has been extended to approximate LQR by avoiding the direct solution of algebraic Riccati equation (ARE); yet, majority of these works do not have favorable scaling properties.

Control and estimation for large-scale systems offers its unique set of challenges due to higher levels of uncertainty, scalability issues, and modeling errors. Nevertheless, model-free synthesis for large-scale systems, as a discipline, is still in its infancy. From a control theoretic perspective, the work [43] addresses some of the aforementioned issues using ideas from mean-field multiagent systems and with the key assumption of partial exchangeability. The work [44], on the other hand, provides a decentralized LQR algorithm based on network consensus that has low complexity, but potentially a high cost of implementation. Finally, semidefinite programming (SDP) projection-based analysis has been examined in [45], where each agent will have a sublinear regret as compared with the best fixed controller in hindsight. The problem has also been considered from a game-theoretic standpoint [46], [47], [48], where agents can have conflicting objectives.

In the following section, we propose our algorithm that iteratively learns necessary control components from a subnetwork that would be used to design a distributed controller for the entire network. We also show that depending on the structure of the problem, not only would our scheme be computationally efficient but also more applicable when model-based control in high dimensions is (practically) infeasible. The structure of our distributed control design is inspired by Borrelli and Keviczky [7], which was subsequently extended to discrete time in [9].

III. MAIN ALGORITHM

In this section, we present and discuss the main algorithm of this article, namely, D2SPI. Given the underlying communication graph $G$, the networked system is considered as a black box, whereas the designer is capable of injecting input signals to the system and observe the corresponding states. The goal of D2SPI is then to find a data-guided suboptimal solution for (3) without the knowledge of system parameters $A$ and $B$. To this end, our approach involves considering the synthesis problem on a subgraph $G_d \subseteq G$, with the associated time-series data. Before presenting the main algorithm, we formalize two useful notions in order to facilitate the presentation.

Definition 1: Given a subgraph $G' \subseteq G$ and a node labeling, let Policy($V_{G'}$) denote the concatenation of policies of the agents in $V_{G'}$, i.e., Policy($V_{G'}$) := $[u_1^T \, u_2^T \, \cdots \, u_i^T]$, where $u_i$ is the feedback control policy of agent $i$ in the subgraph $G'$ as a mapping from $\{x_j | j \in N_i \cup \{i\} \}$ to $\mathbb{R}^m$. Furthermore, we use Policy($V_{G'}$)$_t$ to denote the realization of these policies at time $t$. Similarly, we define State($V_{G'}$) := $[x_1^T \, x_2^T \, \cdots \, x_{|V_{G'}|}]^T$.

The D2SPI algorithm is introduced in Algorithm 1 with the following standard assumption.

Assumption 1: The initial controller $K_1$ is stabilizing for the controllable pair $(A, B)$, and $e_i$ in Algorithm 2 is such that Policy($V_{G'}$)$_t$ remains persistently existing (PE); more precisely, if we collect the state and input difference signals in the vector $\phi_t$, as defined in Algorithm 2, then the PE condition requires that [20] there exists a positive integer $N_0$ and positive constants $\epsilon_0 \leq \tau_0$ such that

$$
\epsilon_0 I \leq 1 \sum_{i=1}^{N} \phi_{t-i} \phi_{t-i}^T \leq \tau_0 I \quad \text{for all} \quad t \geq N_0 \quad \text{and} \quad N \geq N_0.
$$

Remark 1: Note that the controllability of the pair $(A, B)$ and the PE condition are sufficient assumptions to guarantee well-posedness of the data-driven control problem that are commonly adopted in literature [49], [50]. This PE condition is an adaptation of “strong persistent excitation” in [49] to the least-squares problem of recovering unknown parameters of the so-called $Q$-function. The PE condition can be easily satisfied in our setup by ensuring a rich randomness in the signal $e_i$, such as a nondegenerate Gaussian distribution. A more modern treatment of PE condition is stated by a rank condition on the input Hankel matrices in the context of Willems’ fundamental lemma [50], [51]. We also refer to [51] for a discussion on how these assumptions can be relaxed. Furthermore, assuming initial stabilizing controller $K_1$ is also common in the policy iteration literature. For instance, in the case of open-loop stable system $A, K_1$ is simply chosen to be zero. For an unknown and unstable pair $(A, B)$, more elaborate online stabilization techniques have been adopted that are out of the scope of this work. Refer to [52] for one such method.

A. Learning Phase

We refer to the main loop of the algorithm in Line 8 as the learning phase. During the learning phase, we include temporary “auxiliary” links to $G_d$ and make the communication graph a clique. We show such distinction by $G_{d, \text{learn}}$, where $|V_{G_{d, \text{learn}}}| = |V_{G_d}|$, but $G_{d, \text{learn}}$ is a clique. Inherently D2SPI is a policy iteration on $G_{d, \text{learn}}$ that characterizes components $K_k$ and $L_k$, intuitively representing “self” and “cooperative” controls at iteration $k$, respectively. In particular, during the learning phase, we utilize these control components in order to design and update an effective stabilizing controller for the rest of the network $G \setminus G_{d, \text{learn}}$.

We do so by ensuring that during the learning phase, information is exchanged unidirectionally from $G_{d, \text{learn}}$ to the rest of the network; hence, the policy of the agents in $G \setminus G_{d, \text{learn}}$ is dependent on those in $G_{d, \text{learn}}$, and not vice versa. After the learning phase terminates, we remove the temporary links added during the learning phase (reinitialize to the original network topology) and synthesize a suboptimal stabilizing control for the entire network. In the learning phase of D2SPI, we use a
Algorithm 1: Data-Driven Structured Policy Iteration.

1: Initialization (\( t \leftarrow 1, k \leftarrow 1 \))
2: Choose \( G_0 \subseteq \mathcal{G} \) with \( d = d_{\text{max}}(G) + 1 \)
3: Obtain \( Q_1, Q_2, R, \) and set \( Q_d \leftarrow Q_1 + dQ_2 \)
4: Get \( x_1 \in \mathbb{R}^{p \times m} \) and \( H_0 \in \mathbb{R}^{p \times p}, p = d(n + m) \)
5: Set \( P_0 = \beta \delta_{p(p+1)/2} \) for large \( \beta > 0 \) and fix \( \Sigma \)
6: Set \( K_1 \) stabilizing \( (1), L_1 = 0, \Delta K_1 = K_1 \)
7: Turn on temporary links in \( G_d \) and set \( \tau_1 \leftarrow 0 \)

8: While \((K_k, L_k)\) has not converged, do ("learning phase")
9: Set Policy\(_{(V_g)}(V_g)\) such that for each \( i \in V_g \)
   \[
   u_i \leftarrow \Delta K_k x_i + L_k \sum_{j \in N_i} \frac{x_j}{d} x_j, \quad \text{if} \ i \in V_g \setminus V_d
   \]
   \[
   u_i \leftarrow \Delta K_k x_i + L_k \sum_{j \in V_g \setminus V_d} x_j, \quad \text{if} \ i \in V_g \setminus V_d
   \]
10: Evaluate \( H_k \) from Algorithm 2

   \( H_k \leftarrow \text{SPS}(G, G_d, \text{Policy}(V_g), H_{k-1}, P_0) \)
11: Recover \( X_1, X_2, Y_1, Y_2, Z_1, \) and \( Z_2 \) from \( H_k \)
   \[
   X_1 \leftarrow \tilde{H}_k[1 : n, 1 : n]
   \]
   \[
   Y_1 \leftarrow \tilde{H}_k[dn + 1 : dn + m, dn + 1 : dn + m]
   \]
   \[
   Z_1 \leftarrow \tilde{H}_k[dn + 1 : dn + m, 1 : n]
   \]
   \[
   X_2 \leftarrow \tilde{H}_k[1 : n, n + 1 : 2n]
   \]
   \[
   Y_2 \leftarrow \tilde{H}_k[dn + 1 : dn + m, dn + m + 1 : dn + 2m + 1]
   \]
   \[
   Z_2 \leftarrow \tilde{H}_k[dn + 1 : dn + m, n + 1 : 2n]
   \]
   \[
   \Delta X \leftarrow X_1 - X_2, \quad \Delta Y \leftarrow Y_1 - Y_2, \quad \Delta Z \leftarrow Z_1 - Z_2.
   \]
12: Update the control components
   \[
   F^{-1} \leftarrow Y_1 - (d - 1)Y_2 (Y_1 + (d - 2)Y_2)^{-1} Y_2
   \]
   \[
   G \leftarrow (Y_2 + (1 - Y_2)^{-1} Y_1 (Y_1 - Y_2)^{-1} Y_2
   \]
   \[
   K_{k+1} \leftarrow -FZ_1 - (d - 1)GZ_2,
   \]
   \[
   L_{k+1} \leftarrow -FZ_2 + GZ_1 + (d - 2)GZ_2,
   \]
   \[
   \Delta K_{k+1} \leftarrow K_{k+1} - L_{k+1}.
   \]
13: Obtain the stability margin
   \[
   \Xi_{k+1} \leftarrow \Delta X - Q_d + \Delta K_{k+1}^T \Delta Z
   \]
   \[
   + \Delta Z^T \Delta K_{k+1} + \Delta K_{k+1}^T (\Delta Y - R) \Delta K_{k+1}
   \]
   \[
   \gamma_{k+1} \leftarrow \frac{\lambda_{\min}(\Delta K_{k+1}^T R \Delta K_{k+1} + Q_d)}{(1 + \gamma_{k+1})}
   \]
   \[
   \tau_{k+1} \leftarrow \sqrt{(1 + \gamma_{k+1})/(1 + \gamma_{k+1})}
   \]
14: Go to Line 8 and set \( k \leftarrow k + 1 \)
15: Switch OFF the temporary links and retrieve \( G_d \)
16: Set Policy\(_{(V_g)}(V_g)\) such that for each \( i \in V_g \)
   \[
   u_i \leftarrow \Delta K_k x_i + \frac{\tau}{\tau + 1} L_k \sum_{j \in N_i} x_j
   \]

B. Subgraph Policy Evaluation Subroutine

This process is performed in subgraph policy evaluation (SPE) (see Algorithm 2) subroutine by inputting (sub)graph \( G, G_d, \text{learn} \), the mapping policy \( (V_g) \), and the previous estimate of \( H_{k-1} \). As will be discussed in Appendix A-A, \( H_k \) contains the required information to determine the two control components \( K_k \) and \( L_k \) from data. We extract this square matrix through a recursive update on the vector \( \theta_{k-1} \), derived from half-vectorization of \( H_{k-1} \), solving RLS for the linear equation \( R(\tilde{x}_i, \tilde{u}_i) = \zeta_i \theta_{k-1} \), where \( R(\tilde{x}_i, \tilde{u}_i) \) denotes the local cost and \( \zeta_i \in \mathbb{R}^p \) contains the data measurements. We use subscript \( k \) for policy update and \( t \) for data collection. The adaptive nature of the algorithm involves the exploration signal \( e_t \) to be augmented to the policy vector in order to provide the persistence of excitation.

C. Persistence of Excitation and Convergence of SPE

In our setup, \( e_t \) is sampled from a normal distribution \( e \sim \mathcal{N}(0, \Sigma) \), where the choice of the variance \( \Sigma > 0 \) is problem specific. In practice, excitation of the input is a subtle task and has been realized in a variety of forms such as random noise [19], sinusoidal signals [53], and exponentially decaying noise [54]. We denote by \( \mathcal{P} \) the projection factor that is reset to \( P_0 > 0 \) for each iteration. Convergence of SPE—guaranteed based on the persistence of excitation condition—is followed by the update of \( H_k \) that encodes the necessary information to obtain \( K_k \) and \( L_k \).

D. Learning Control Components Using Patterned Monoid

Learning the control components \( K_k \) and \( L_k \) is achieved by first recovering the block matrices \( X_1, X_2, Y_1, Y_2, Z_1, \) and \( Z_2 \) from \( H_k \) that are further utilized to form intermediate variables \( F \) and \( G \). Matrix inversions on line 12 of Algorithm 1 will be justified in Section IV and Lemma 2. Such recovery of meaningful blocks from \( H_k \) is due to the specific structure resulting from adding extra links to \( G_d \) that is captured systematically by the Patterned monoid; this point will be discussed further subsequently in Proposition 2.

E. Learning Gain Margin and the Distributed Feedback Design

Each iteration loop is completed by updating the parameters \( \gamma_k \) and \( \tau_k \) that prove instrumental in the stability analysis of the proposed distributed controller for the entire network. Finally, with the convergence of D2SPI, \( G_d, \) is retrieved by removing the temporary links and the structured policy is extended to the entire graph \( G \) in line 16.

Let us point out a few remarks on the computational complexity of the proposed algorithm. First, note that the inverse operations on line 12 occur on matrices of size \( m \times m \) and, hence, computationally inexpensive. Furthermore, the complexity of finding extreme singular values—as on line 13 in Algorithm 1—is known to be \( O(n^2) \) [55]. Hence, the computational complexity of D2SPI is mainly due to the SPE recursion that is equivalent
to the complexity of RLS for the number of unknown system parameters in \( G_d \), i.e., the computational cost is \( O(d^3(n + m)^2) \) \[56\]. This implies that the computational complexity of the algorithm is fixed for any number of agents \( N \), as long as the maximum degree of the graph retains its order.

**Remark 2:** Adding temporary links within the subgraph \( G_d \) is an effective means of learning optimal \( K_k \) and \( L_k \) for the subgraph \( G_{d,\text{learn}} \) by utilizing dynamical interdependencies among the agents. Although initializing \( K_k \) such that (1) is Schur stable is a standard assumption in data-driven control, obtaining this initial gain for an unknown system is nontrivial. While we invoke this assumption in this work, the interested reader is referred to \[40\] and \[57\] for more recent works pertaining to this assumption and related system-theoretic intricacies \[51\].

### IV. Convergence and Stability Analyses

In this section, we provide convergence and stability analyses for the D2SPI algorithm. First, we study the structure and stability margins of each local controller and proceed to establish stability properties of the proposed controller for the entire network throughout the learning process. Finally, we show the convergence of D2SPI to a stabilizing suboptimal distributed controller followed by the derivation of a suboptimality bound characterized by the problem parameters. For clarity, we defer some of the subtleties of the analysis and detailed proofs to Appendix A.

First, let us demonstrate how a specific structure and stability of the controller for the subgraph \( G_{d,\text{learn}} \), when properly initialized, can be preserved throughout the D2SPI algorithm.

**Proposition 2:** Let \( \tilde{K}_k := I_d \otimes (K_k - L_k) + I_d^\top \otimes L_k \) for all \( k \geq 1 \), with \( K_k \) and \( L_k \) as in Algorithm 1. Under Assumption 1 and throughout the learning phase (for all \( k \geq 1 \), \( \tilde{K}_k \) is stabilizing for the system in \( G_{d,\text{learn}} \) and Policy \( \nu_{G_{d,\text{learn}}} \)) is \( \tilde{K}_k \) State \( \nu_{G_{d,\text{learn}}} \) for all \( t \). Furthermore, \( \Delta K_k := K_k - L_k \) stabilizes the dynamics of a single agent, i.e., \( A + B \Delta K_k \) is Schur stable.

Note that Proposition 2 proves the existence of a stabilizing controller \( \Delta K_k \) and its corresponding cost-to-go matrix \( \Delta P_k \). In the following, our goal is to design a suboptimal distributed controller for the entire networked system on \( G \) based on the components that shape \( \Delta K_k \). This extension is built upon the stability margin derived next.

**Proposition 3:** At each iteration \( k \geq 1 \), let \( K_k, L_k \), and \( \tau_k \) be obtained via Algorithm 1. Then, \( A + B(K_k - L_k) \) is Schur stable for all \( \alpha \) satisfying \( |\alpha - 1| \leq \tau_k \).

The stability margin \( \tau_k \) in Proposition 3 is upper-bounded by the stability margin of the pair \( (A + B(K_k - L_k), B) \). This implies that if the original closed-loop system for an agent does not have a favorable stability margin, then \( \tau_k \) can be small—reducing the influence of the agent’s neighbors on its policy (line 16 of Algorithm 1). Nonetheless, Proposition 3 provides model-free stability gain margins \( \tau_k \) at each iteration of the algorithm for the dynamics of a single agent in \( G \). In our analysis, we take advantage of these margins to characterize stability guarantees for the controller proposed during the learning phase. This is captured in the following result.

**Theorem 1:** Suppose that \( K_k, L_k, \) and \( \tau_k \) are defined as in Algorithm 1. Then, under Assumption 1, the control policy Policy\( \nu_k \) designed during the learning phase (line 9) stabilizes the network \( G \) at each iteration of the learning phase and for any choice of \( \nu_{G_d} \).

Theorem 1 establishes that the proposed feedback mechanism stabilizes the entire network, facilitating control of agents outside of \( G_d \) during the learning phase. In the meantime, the practicality and suboptimality of the algorithm depend on its convergence addressed next.

**Theorem 2:** Under Assumption 1 and (long enough) finite termination of Algorithm 2, Algorithm 1 converges, i.e., \( K_k \to K^*, L_k \to L^* \), and \( \tau_k \to \tau^* \) as \( k \to \infty \), where \( K^* = I_d \otimes (K^* - L^*) + I_d^\top \otimes L^* \) is the optimal solution to the infinite-horizon state-feedback LQR problem with system parameters \( (A, B, Q, R) \) defined as \( A = I_d \otimes A, B = I_d \otimes B, Q = I_d \otimes (Q_1 + dQ_2) - I_d^\top \otimes Q_2, \) and \( R = I_d \otimes R \).

Finally, we note that as the temporary links introduced during the learning phase are removed, the structure of the agents’ interaction is once again the original network \( G \). As such, it is vital to provide stability guarantees after Algorithm 1 terminates and components of the control design have converged. This issue is addressed in the following corollary whose proof is similar to that of Theorem 1 and thus omitted.

**Corollary 1:** Suppose that \( K^*, L^*, \gamma^* \), and \( \tau^* \) are given as in Theorem 2 under a convergent Algorithm 1. Then, Policy\( \nu_k \) defined on line 16) stabilizes the entire networked system in (2).

We conclude this section by exploring the suboptimality of the proposed policy. Given the problem parameters, let \( K_{\text{struc}} \) denote the globally optimal distributed solution for the structured LQR problem in (3) with the associated cost matrix \( P_{\text{struc}} \). Given any

**Algorithm 2:** Subgraph Policy Evaluation.

1. **Input:** Graph \( G \), subgraph \( G' \subseteq G \), Policy\( \nu_{G'} \), \( H \), \( \mathcal{P} \)
2. **Output:** Updated cost matrix \( H^+ \) associated with \( G' \)
3. **While \( H \) has not converged, do**
   4. Set \( \hat{x}_t := \text{State}(\nu_{G'})_t \) and \( \hat{u}_t := \text{Policy}(\nu_{G'})_t \)
   5. Choose \( e_t \sim N(0, \Sigma) \) and update Policy\( \nu_{G'} \) as \( \hat{u}_t \leftarrow \hat{u}_t + e_t \) for all \( t \in \nu_{G'} \)
   6. Run \( G \) under policy Policy\( \nu_{G'} \)
   7. Collect State\( \nu_{G'} \) only from \( G' \) and set \( \tilde{x}_{t+1} := \text{State}(\nu_{G'})_{t+1} \), \( \tilde{u}_{t+1} := \text{Policy}(\nu_{G'})_{t+1} \)
   8. Compute \( \zeta_t = \text{vech}(\phi_t \phi_t^\top) \) and \( R_t(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_t^\top (I \otimes Q_d - I_d^\top \otimes Q_2) \tilde{x}_t + \tilde{u}_t^\top (I \otimes R) \tilde{u}_t \)
   9. Set \( \theta := \text{vech}(H) \) and update \( \theta := \mathcal{P}_t \zeta_t + \mathcal{P_t} \zeta_t^\top \theta + \mathcal{P_t} \zeta_t^\top \mathcal{P_t} \)
   10. Find \( H^+ := \text{vech}^{-1}(\theta) \), update \( H := H^+ \) and \( t := t + 1 \)
other stabilizing structured policy \( \hat{K} \) associated with cost matrix \( \hat{P} \), we define the optimality gap as
\[
\text{gap}(\hat{K}) := \text{tr}[\hat{P} - \hat{P}_{\text{struc}}^*].
\]
The following theorem provides an upper bound on the optimality gap of structured policy learned by D2SPI based on the problem parameters. In particular, when the system is “contractible,” the derived upper bound depends on the difference of the distributed controller with that of unstructured optimal LQR controller.

**Theorem 3:** Let \( \hat{K}^* \) be the structured policy learned by Algorithm 1 at convergence, corresponding to the cost matrix \( \hat{P}^* \). Moreover, let \( \hat{K}_{\text{struc}} \) denote the optimal (unstructured) solution to the infinite-horizon state-feedback LQR problem with parameters \((A, B, Q, R)\) with the cost matrix \( \hat{P}_{\text{struc}} \). If \( \hat{A}_{\hat{K}_{\text{struc}}} = \hat{A} + \hat{B} \hat{K}_{\text{struc}} \) is contractible, then
\[
0 \leq \text{tr}(\hat{K}^*) \leq \text{tr}(M)/|1 - \sigma^2_{\max}(\hat{A}_{\hat{K}_{\text{struc}}})|,
\]
where \( M := (R + \hat{B}^\top \hat{P} B)(\hat{K}^* - \hat{K}_{\text{struc}}) + 2 \hat{A}^\top \hat{P} \hat{B} \hat{K}^* \).

**Remark 3:** First, recall how the converged policy by Theorem 2 is related to the optimal LQR policy on the fully connected subgraph \( G_{\text{dlearn}} \). Second, the optimality gap is characterized by \( \text{tr}(M) \), which is essentially proportional to the difference \( \hat{K}^* - \hat{K}_{\text{struc}} \), i.e., how close our designed structured policy is to the unstructured LQR that directly depends on the connectivity of the graph topology. Third, the contractibility of the pair \((A, B)\) is more restrictive condition than regularizability of the system [40], a notion that has recently been employed in iterative data-guided control methods [58], [59]. Contractibility also facilitates the validity of assuming access to the initial stabilizing controller.

**V. SIMULATION EXAMPLES**

In this section, we examine the performance and convergence of D2SPI. In order to assess the suboptimality of the synthesized controllers, we report the trace of cost matrices, \( \text{tr}(\hat{P}_k) \), associated with the proposed distributed controller learned by D2SPI at iteration \( k \). As the optimal distributed design is unknown, we compare these results against the optimal cost for the unconstrained LQR problem, \( \text{tr}(P^*_{\text{LQR}}) \), obtained via the solution of the ARE with parameters \((A, B, Q, R)\). Note that this is an infeasible solution to the problem in (3); nevertheless, it provides a theoretical lower bound to evaluate the performance of any feasible solution—including the optimal one.\(^5\)

**A. Convergence—Randomly Selected Parameters**

In the first example, we sample continuous-time system parameters of a single agent \((A, B)\) from a zero-mean normal distribution with unit covariance, such that \( A \in \mathbb{R}^{5\times 5} \) and \( B \in \mathbb{R}^{5\times 3} \). We then consider a path graph of ten agents and demonstrate how Algorithm 1 converges for this network using different instances of the system parameters. The continuous-time system dynamics of a single agent is discretized with a sampling rate of \( \Delta T = 0.1 \) s. We set \( Q_1 = 0.2I_5 \), \( Q_2 = I_5 \), and \( R = I_3 \) in (3). We assume a random exploration signal sampled from a normal distribution \( \epsilon_t \sim \mathcal{N}(0, \sigma^2) \), where the variance \( \sigma^2 \) is chosen accordingly for different input channels.

**B. Network of Homogeneous Plants**

In this example, we apply D2SPI to two other simulation scenarios involving homogeneous networks of agents with unknown and unstructured model uncertainties. In particular, we use the dynamics of plants with continuous-time system parameters \((A, B)\) (as reported in [60, Appendix F]), in conjunction with random \( d \)-regular graph topologies of different sizes. We then examine the efficacy of D2SPI by illustrating the cost associated with the proposed distributed controller as a function of nodes in the graph. The rest of the problem parameters are chosen identical to the setup in Section V-A. Uncertainty in the model is introduced in this example as follows. Each agent follows an unknown linear time-invariant dynamics similar to (1) with \( A \) replaced by \( A + \Delta A \), where entries of \( \Delta A \) are sampled from a normal distribution \( \mathcal{N}(0, 0.05) \). Assuming that one has access to a stabilizing controller \( K_1 \) for the system with nominal parameters \( A \) and \( B \), we set the initial stabilizing controller to be the LQR optimal controller with parameters \((A, B, Q_1, R)\).

Fig. 2 shows the results of the second simulation example, illustrating how the cost of the proposed controller changes with \( R = I_3 \) in (3). We assume a random exploration signal sampled from a normal distribution \( \epsilon_t \sim \mathcal{N}(0, \sigma^2) \), where the variance \( \sigma^2 \) is chosen accordingly for different input channels.

\(^5\) All the simulations were run on a 3.2-GHz quad-core Intel Core i5 CPU and in MATLAB. The scripts are publicly available at https://github.com/shahriarta/D2SPI.
respect to the number of nodes in a path graph with different number of nodes. Fig. 3(a) compares the cost associated with our design $\tilde{P}_\infty$ against the cost of the initial controller $P_1$ and the (infeasible) LQR controller $P_{LQR}$. Fig. 3(b) illustrates the evolution of the normalized suboptimality of our proposed algorithm (with respect to the infeasible LQR controller) as a function of number of nodes in the corresponding graphs. Fig. 3(c) and (d) shows similar results for the random 3-regular graph topologies with even number of nodes.

As can be validated from Fig. 3, the cost associated with our final proposed distributed controller has significantly improved the optimality of the initial controller. In particular, the normalized suboptimality errors of our final design are less than 6% and 2.4% for path graph and random 3-regular graph topologies, respectively. Furthermore, this normalized error generally decreases as the number of nodes in the corresponding graphs increases.

VI. CONCLUDING REMARKS

In this article, we have proposed the D2SPI algorithm as an efficient model-free distributed control synthesis process for potentially high-dimensional network of homogeneous linear systems. D2SPI is built upon a construction referred to as Patterned Monoid that facilitates exploiting network symmetries for policy synthesis consistent with the underlying network. Such symmetries allow a data collection procedure during the learning phase (with temporary additional links) for a smaller portion of the network. Using data collection on this smaller subnetwork, we are then able to synthesize a distributed feedback mechanism for the entire system—even during the learning phase. Moreover, D2SPI builds upon parameter estimation techniques that represent an end-to-end policy prediction directly from the observed data. Extension of the setup proposed for D2SPI involves heterogeneous networked systems, which is currently being examined as part of our future work.

APPENDIX A

ANALYSIS AND PROOFS

In this section, we provide the building blocks needed for the proofs and analysis of our algorithm. We first provide some insights on how the setup is connected to the classic model-based LQR machinery and some previously established results that we leverage from the literature. The main proofs then follows.

A. Underlying Model of the Subsystem $G_d$

The configuration of the synthesis problem in D2SPI intertwines an online recursion on the subsystem corresponding to $G_d$ and the original system $G$. In particular, during the learning phase, considering the same cost structure and problem parameters as in (3)—but for the completed subgraph $G_{d,learn}$—results in the $(\tilde{A}, \tilde{B}, \tilde{Q}, \tilde{R})$ parameters, as defined in Theorem 2. Then, similarly from (1), the dynamics of the subgraph $G_{d,learn}$ assumes the form

$$\tilde{x}_{t+1} = \tilde{A} \tilde{x}_t + \tilde{B} \tilde{u}_t \quad (4)$$

where $\tilde{x}_t$ and $\tilde{u}_t$ are formed from the concatenation of state and control signals in $G_{d,learn}$—recall that $\tilde{u}_t$ is also denoted by Policy($V_{G_{d,learn}}$, $\tilde{z}_t$) in the algorithm to emphasize the temporal implementation of a specific policy in Algorithm 2. From the Bellman equation [61] for the LQR problem with these parameters, the cost matrix $\tilde{P}_k$ of $G_{d,learn}$ is correlated with the one-step LQR cost as

$$\tilde{x}_t^T \tilde{P}_k \tilde{x}_t = \mathcal{R}(\tilde{x}_t, \tilde{u}_t) + \tilde{x}_{t+1}^T \tilde{P}_k \tilde{x}_{t+1} \quad (5)$$

where $\mathcal{R}(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_t^T \tilde{Q} \tilde{x}_t + \tilde{u}_t^T \tilde{R} \tilde{u}_t$ and $\tilde{P}_k$ satisfies the Lyapunov equation

$$\tilde{P}_k = (\tilde{A} + \tilde{B} \tilde{K}_k)^T \tilde{P}_k (\tilde{A} + \tilde{B} \tilde{K}_k) + \tilde{Q} + \tilde{K}_k^T \tilde{R} \tilde{K}_k \quad (6)$$

and $\tilde{K}_k$ is the controller policy at iteration $k$. The dynamic programming solution to the LQR problem suggests a linear feedback form $\tilde{u}_t = \tilde{K}_k \tilde{x}_t$ for the subsystem $G_{d,learn}$ at each iteration. Combining (4) and (5) with some rearrangements results in

$$\tilde{x}_t^T \tilde{P}_k \tilde{x}_t = \tilde{z}_t^T \begin{bmatrix} \tilde{Q} + \tilde{A}^T \tilde{P}_k \tilde{A} & \tilde{A}^T \tilde{P}_k \tilde{B} \\ \tilde{B}^T \tilde{P}_k \tilde{A} & \tilde{R} + \tilde{B}^T \tilde{P}_k \tilde{B} \end{bmatrix} \tilde{z}_t$$

$$=: \tilde{z}_t^T \begin{bmatrix} \tilde{H}_k |_{11} & \tilde{H}_k |_{12} \\ \tilde{H}_k |_{21} & \tilde{H}_k |_{22} \end{bmatrix} \tilde{z}_t = \tilde{z}_t^T \tilde{H}_k \tilde{z}_t \quad (7)$$

where $\tilde{z}_t = [\tilde{x}_t^T \tilde{u}_t]^T$. Then, the following policy update (due to Hewer) is guaranteed to converge to the optimal LQR policy under the controllability assumption [21]

$$\tilde{K}_{k+1} = - \left( \tilde{R} + \tilde{B}^T \tilde{P}_k \tilde{B} \right)^{-1} \tilde{B}^T \tilde{P}_k \tilde{A} = -[\tilde{H}_k |_{22}^{-1} \tilde{H}_k |_{21}] \quad (8)$$
which is also reconstructed by information in $\bar{H}_k$. Furthermore, the cost matrix in (6) can also be reconstructed by the same information

$$\bar{P}_k = [\bar{H}_k]_{11} + [\bar{H}_k]_{12} \bar{K}_k + \bar{K}_k [\bar{H}_k]_{21} + \bar{K}_k [\bar{H}_k]_{22} \bar{K}_k.$$  

(9)

Hence, $\bar{H}_k$ provides the required information to perform both policy update (8) and policy evaluation steps (9) in a policy iteration algorithm. We will see that because of the particular structure of our setup, $\bar{H}_k$ enjoys a special block pattern captured by the proposed Patterned monoid, justifying the recovery of the block matrices $X_1, X_2, Y_1, Y_2, Z_1, \text{and } Z_2$ from $\bar{H}_k$ in line 11 of D2SPI. D2SPI leverages this idea to implicitly learn $\bar{H}_k$ from data (by adapting the idea of [19]) and exploit these matrix blocks in order to find an efficient suboptimal solution to the main distributed problem in (3). Here, in addition to the policy update from data as in (8), we show that the same information can be used to also learn a gain margin directly from data (see Proposition 3). This gain margin is then used to guarantee the stability of the entire network (see Theorem 1).

Finally, for technical reasons, recall that the infinite-horizon state-feedback LQR problem with parameters $(A, B, Q, R)$ can be cast as the minimization of

$$f_r(K) := \text{tr}[P_r \Sigma] \quad (10)$$

over the static stabilizing policy $K$, for some initial state distribution with covariance $\Sigma > 0$, where $P_r$ is cost matrix associated with $K$ satisfying the following Lyapunov equation [16], [17], [62]:

$$P_r = (A + B K)^T P_r (A + B K) + Q + K^T R K.$$

Herein, we set $\Sigma = I$ and consider $f_r(K)$.

### B. Technical Observations and Main Proofs

In the remainder of this section, we first restate some well-known technical facts to make this article self-contained and then propose a few additional algebraic facts for our analysis whose proofs are deferred to Appendix B. We then continue with the proof of the main results.

**Lemma 1:** The following relations hold:

1) (see [63]) When $X > 0$

$$M^T X N + N^T X M \geq -\left(a M^T X M + \frac{1}{a} N^T X N\right)$$

$$M^T X N + N^T X M \leq a M^T X M + \frac{1}{a} N^T X N$$

where $M, N \in \mathbb{R}^{n \times m}$ with $n \geq m$ and $a > 0$.

2) (see [64, Lyapunov equation]) Suppose that $A \in \mathbb{R}^{n \times n}$ has spectral radius less than 1, i.e., $\rho(A) < 1$. Then, $A^T X A + Q - X$ has a unique solution, and $X = \sum_{j=0}^{\infty} (A^T)^j Q A^j$. In this case, if $Q > 0$, then $X > 0$.

3) (see [63, Block matrix inverse formula (0.8.5.6)]) The following identity holds for matrices $A, B, C,$ and $D$ with compatible dimensions:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} H^{-1} & -H^{-1} BD^{-1} \\ -D^{-1} CH^{-1} & D^{-1} + D^{-1} CH^{-1} BD^{-1} \end{bmatrix}$$

where $D$ and $H = A - BD^{-1} C$ are invertible.

4) (Matrix inversion lemma [65]) The following identity holds:

$$(A + U C V)^{-1} = A^{-1} - A^{-1} U (C^{-1} + V A^{-1} U)^{-1} V A^{-1}$$

for matrices $A, U, C,$ and $V$ with compatible dimensions where $A, C,$ and $A + U C V$ are invertible.

Finally, we provide the main technical lemma that streamlines the properties of the Patterned monoid under algebraic manipulation, which will be frequently used in the proofs of Proposition 2 and Theorem 2.

**Lemma 2:** Suppose $N_r \in \mathbb{P}M(r \times n, \mathbb{R})$ for some $n$ and $r \geq 2$ such that $N_r = I_r \otimes \begin{pmatrix} A - B \end{pmatrix} + \mathbb{I}_r \mathbb{I}^T \otimes B$, for some $A \in GL(n, \mathbb{R}) \cap \mathbb{S}^{n \times n}$ and $B \in \mathbb{S}^{n \times n}$. Then, the following hold:

1) $\text{det}(N_r) = \text{det}(A - B)^{-1} \text{det}(A + (r - 1) B)$.

2) If $N_r > 0$, then we have $A + (A - B) > 0$ for all $\ell = 0, 1, \ldots, r$. Furthermore, $A - B (A + (A - B))^{-1} B$ is invertible for $\ell = 1, 2, \ldots, r - 1$.

3) If $N_r > 0$, then $N_r^{-1} \in \mathbb{P}M(r \times n, \mathbb{R})$; in fact

$$N_r^{-1} = I_r \otimes \left( F_r + G_r \right) - \mathbb{I}_r \mathbb{I}^T \otimes G_r$$

with $F_r$ and $G_r$ defined as

$$F_r = \left( A - (r - 1) B (A + (r - 2) B)^{-1} B \right)^{-1}$$

$$G_r = \left( A + (r - 1) B \right)^{-1} (A - B)^{-1}.$$

4) If also $M_r \in \mathbb{L}(r \times n, \mathbb{R})$, i.e., $M_r = I_r \otimes (C - D) + \mathbb{I}_r \mathbb{I}^T \otimes D$ for some $C, D \in \mathbb{R}^{n \times n}$, then

$$N_r M_r = I_r \otimes \begin{pmatrix} A - B \end{pmatrix} (C - D) + \mathbb{I}_r \mathbb{I}^T \otimes (B - C) + (A - B) D + r B D.$$
Define the Lyapunov candidate

\[ \Delta_{k+1} = 2 \tilde{c} - \tilde{c} \in \mathbb{R}^d \otimes \mathbb{R}^d \]

and

\[ Q = \Delta_{k+1} R \Delta_{k+1} \otimes \mathbb{R}^d \]

Thus, the latter bound can be obtained completely from data as

\[ \lambda_{\text{max}}(\Gamma_k) \leq ((1 - \alpha) a - \gamma_k) \lambda_{\text{max}}[\Xi_k + L_k^T(\Delta Y - R)L_k] \]

with

\[ \gamma_k \coloneqq \min_{\beta} \{ Q_d + \Delta K_k^T R \Delta K_k \} \]

which coincides with updates in line 13 of Algorithm 1.

Finally, from the hypothesis $1 - \tau_k < \alpha < 1$, we obtain $\gamma_k^2 < 4 \beta^2 \gamma_k - 4 \beta^2 > 0$. However, since $\gamma_k > 0$, this second-order term in $\gamma_k$ is positive only if

\[ \gamma_k > 2 \beta^2 + 2 \beta \sqrt{\beta^2 + 1} > 2 \beta (\beta + \sqrt{\beta^2 + 1}) = (1 - \alpha) a. \]

Therefore, $\Delta V_k(x_t) < 0$ for $1 - \tau_k < \alpha < 1$. Similar reasoning for $1 < \alpha < 1 + \tau_k$ also shows that $\Delta V_k(x_t) < 0$, which completes the proof.

**4) Proof of Theorem 1**: From Definition 1 (according to a consistent choice of labeling of the nodes so that the last $d$ nodes are chosen as $G_d$) and Proposition 2, the feedback policy in (line 9) of Algorithm 1 can be cast in the compact form

\[
\text{Policy}_{k}(\mathcal{G}) = \begin{bmatrix} \text{Policy}_{k}(V_{\mathcal{G}|d,d_{\text{learn}}}) \\ \text{Policy}_{k}(V_{\mathcal{G}|d_{\text{learn}}}) \end{bmatrix} = \begin{bmatrix} \tilde{K}_k \tilde{L}_d \tilde{c} \\ \tilde{G} \end{bmatrix} \begin{bmatrix} \text{State}_{k}(V_{\mathcal{G}|d,d_{\text{learn}}}) \\ \text{State}_{k}(V_{\mathcal{G}|d_{\text{learn}}}) \end{bmatrix}
\]

Note that the structure of $\tilde{K}_k$ emanates from the fact that the information exchange is unidirectional during the learning phase. Now, consider the closed-loop system of $\mathcal{G}$ as

\[
\tilde{A}_{\mathcal{G}|d} = \tilde{A} + \tilde{B} \tilde{K}_k
\]

where $\tilde{K}_k = I_d \otimes (K_k - L_k) + I_d \otimes \tilde{L}_d \otimes L_k$, $\tilde{A}_{\mathcal{G}|d} = I_{N-d} \otimes K - (I_{N-d} - \tilde{A}_{\mathcal{G}|d}) \otimes L_k$, $A_{\mathcal{G}|d}$ denotes the adjacency matrix of $\mathcal{G}$, and $\tilde{A}_{\mathcal{G}|d}$ is its submatrix capturing the interconnection of $\mathcal{G} \setminus \mathcal{G}_d$ and $\mathcal{G}_d$.
such that \( TST^{-1} = J \). Now, consider the following similarity transformation of \( \tilde{A}_{cl} \):

\[
\begin{bmatrix}
T \otimes I_n & 0 \\
0 & I_d \otimes I_n
\end{bmatrix}
\begin{bmatrix}
\tilde{A}_{cl} & 0 \\
0 & \tilde{A}_{cl}
\end{bmatrix}
\begin{bmatrix}
T \otimes I_n & 0 \\
0 & I_d \otimes I_n
\end{bmatrix}^{-1}
= \begin{bmatrix}
I_n \otimes A + B(K_k - \lambda_i(S)L_k) & J \otimes BL_k \\
0 & \tilde{A}_{K_k}
\end{bmatrix}.
\]

Note that \( I_n \otimes A + B(K_k - \lambda_i(S)L_k) \) is a block upper triangular matrix whose diagonal blocks are equal to \( A + B(K_k - \lambda_i(S)L_k) \) for \( i = 1, \ldots, N - d \). We already know from Proposition 2 that \( A_{cl} \) is Schur stable. Hence, in order to show that \( \tilde{A}_{cl} \) is Schur stable, it suffices to show that \( \rho(A + B(K_k - \lambda_i(S)L_k)) < 1 \) for \( i = 1, \ldots, N - d \). Recall that \( |\lambda_i(A_{cl})| \leq d_{max} \) [66]; thus, by definition of \( S \) and the fact that \( d_{max} = d - 1 \), we conclude that \( |\lambda_i(S) - 1| \leq \tau_k \). The claim now follows directly from Proposition 3.

5) Proof of Theorem 2: At iteration \( k \) of the learning phase in Algorithm 1, consider \( H_k \) and its corresponding blocks, as defined in (7). First, we consider the structure of the stabilizing feedback policy \( K_k \), as shown in Proposition 2, together with that of system parameters \( (\hat{A}, \hat{B}, \hat{Q}, \hat{R}) \), and apply Lemma 2 and Proposition 1 to conclude that \( H_{k,11}, H_{k,22} \in PM(d \times n, \mathbb{R}) \) and \( H_{k,21} \in L(d \times n, \mathbb{R}) \). Thus, we get

\[
H_{k,11} = I_d \otimes (X_1 - X_2) + 11^T \otimes X_2
\]
\[
H_{k,22} = I_d \otimes (Y_1 - Y_2) + 12^T \otimes Y_2
\]
\[
H_{k,21} = I_d \otimes (Z_1 - Z_2) + 11^T \otimes Z_2
\]

which coincides with the recovery of \( X_i, Y_i, Z_i \) for \( i = 1, \ldots, 11 \) of Algorithm 1. We can also unravel the structure of block matrices constructing \( H_k \) and obtain that

\[
X_1 = Q_1 + (d - 1)Q_2 + A^TP_1A, \quad Y_2 = B^TP_2B
\]
\[
X_2 = -Q_2 + A^TP_2A, \quad Z_1 = B^TP_1A
\]
\[
Y_1 = R + B^TP_1B, \quad Z_2 = B^TP_2B
\]

with \( P_1 \) and \( P_2 \) as in (11). Similarly, by Lemma 2, we obtain

\[
H_{k,22} = I_d \otimes (F - G) + 12^T \otimes G
\]

where (by item 3 in Lemma 2) \( F \) and \( G \) must satisfy

\[
F = Y_1 - (d - 1)Y_2Y_1 + (d - 2)Y_2^{-1}Y_2
\]
\[
G = (Y_1 + (d - 1)Y_2^{-1}Y_2(Y_1 - Y_2)^{-1}
\]

which coincides with the definitions in line 12 of Algorithm 1. Finally, by definition of \( K_{k+1} \) and \( L_{k+1} \) in line 12, and item 4 in Lemma 2, one can verify that \( K_{k+1} = -[H_{k,22}]^{-1}H_{k,21} \), which coincides with the policy iteration in Heuer’s algorithm [21] for the system in \( G_{d,learn} \) (see also Appendix A-A). Note that by assumption, the pair \( (A, B) \) is controllable, so is the system \( (\hat{A}, \hat{B}) \) in \( G_{d,learn} \). Therefore, these updates are guaranteed to remain stabilizing and converge to the claimed optimal LQR policy \( K^* \), provided that we have access to the true parameters \( H_k \).

Next, consider the LQR cost \( f_1(K^*) \) as in (10) but for the problem parameters \( (\hat{A}, \hat{B}, \hat{Q}, \hat{R}) \). For completing the proof of convergence, it is left to argue that there exists a large enough integer \( C \) such that, at each iteration \( k \) of the learning phase, the RLS in Algorithm 2 provides a more accurate estimation, denoted by \( \hat{\theta}_{k} \), of the true parameters \( \theta_k = \text{vech}(H_k) \), and the LQR cost \( f_1(K^*) = \text{tr}(P_{k+1}) \) decreases. This claim essentially follows by [19, Th. 1], which we try to summarize for completeness. For that, at iteration \( k \), let us denote the policy obtained using the estimated parameters by \( \tilde{K}_k \), which, in turn, estimates the true policy \( K^* \). We then define a “Lyapunov” function candidate

\[
s_k := f_1(\tilde{K}_k) + \|\theta_{k-2} - \theta_{k-1}\|.
\]

Following the same induction reasoning as that of [19, Th. 1] and under PE input, there exists an integer \( C \) such that

\[
s_{k+1} = s_k - \varepsilon_1(C)\|\theta_{k-1} - \theta_{k-2}\| - \varepsilon_2(C)\|\tilde{K}_k - \tilde{K}_{k-1}\|^2
\]

for some positive constants \( \varepsilon_1(C) \) and \( \varepsilon_2(C) \) that are independent of \( k \). Then, \( s_{k+1} \leq s_k \) and

\[
\varepsilon_1(C) \sum_{k=2}^{\infty} \|\theta_{k-1} - \theta_{k-2}\| \leq s_1;
\]
\[
\varepsilon_2(C) \sum_{k=2}^{\infty} \|\tilde{K}_k - \tilde{K}_{k-1}\|^2 \leq s_1.
\]

Also, \( s_1 \) is bounded as \( \tilde{K}_k \) stabilizes \( (\hat{A}, \hat{B}) \). This guarantees that, first, \( \tilde{K}_k \) remains stabilizing as \( f_1(\tilde{K}_k) \leq s_0 \) and \( \hat{Q} > 0 \); second, the estimates \( \hat{\theta}_{k-1} \) become more accurate; and third, \( \tilde{K}_k \) converges to \( K^* \) as \( \tilde{K}_{k+1} \rightarrow K^* \) at \( k \rightarrow \infty \) [21]. This completes the proof.

6) Proof of Theorem 3: Consider the LQR cost \( f_1(K^*) \) as in (10) but for the problem parameters \( (\hat{A}, \hat{B}, \hat{Q}, \hat{R}) \). In the “unstructured” case (i.e., ignoring the constraint \( K \in \mathcal{U}(G_{m, n}(G)) \), we know that the optimal LQR cost matrix for the entire networked system satisfies [67]

\[
\hat{P}_{K} = \hat{A}_{K}^T\hat{P}_{K}\hat{A}_{K} + \hat{K}_l^T R \hat{K}_l + \hat{Q}
\]

where \( \hat{P}_{K} = \arg\min_K f_1(K) \). Moreover, the cost matrix \( \hat{P}_{K} \) associated with the structured policy \( K^* \) learned by Algorithm 1 satisfies

\[
\hat{P}_{K} = \hat{A}_{K}^T \hat{P}_{K} \hat{A}_{K} + \hat{K}_l^T R \hat{K}_l + \hat{Q}
\]

where \( \hat{K}_l \in \mathcal{U}(G_{m, n}(G)) \). Finally, let \( \hat{K}_l \in \mathcal{U}(G_{m, n}(G)) \) denote a “structured” stabilizing optimal LQR policy, which is associated with the cost matrix \( \hat{P}_{K} \). We know that such a policy exists since the smooth cost is lower-bounded and \( \hat{K}_1 = I_N \otimes K_1 \in \mathcal{U}(G_{m, n}(G)) \) is a feasible point of this optimization—as \( K_1 \) is assumed to be stabilizing for the single pair \( (A, B) \). Therefore

\[
\text{tr} [\hat{P}_{K}] = f_1(\hat{K}_l) \leq f_1(\hat{K}_l) \leq f_1(\hat{K}_l) = \text{tr} [\hat{P}_{K}^*]
\]

where the last inequality above follows by the fact that \( \hat{K}_l \) is a feasible solution to the structured problem by construction.
i.e., Κ∗ ∈ U_{m,n}^{N}(G). Therefore, 0 ≤ gap(Κ∗) ≤ tr[Π∗ − Π_{lqr}].
But then, one can obtain from (14), (15), and some algebraic manipulation
that

\[ \hat{Π}^∗ − Π_{lqr} = A_{K_{lqr}}^T (\hat{Π}^∗ − Π_{lqr}) A_{K_{lqr}} + M' \]

where

\[ M' = A_{K}^T \hat{Π}^∗ A_{K} − A_{K_{lqr}}^T \hat{Π}^∗ A_{K_{lqr}} + Κ^∗ \hat{R} Κ^∗ − Κ_{lqr}^T \hat{R} Κ_{lqr} \]

Since \( \hat{Π}^∗ − Π_{lqr} \geq 0 \) and \( A_{K_{lqr}} \) is contractible by the hypothesis,
from the first part and [68, Th. 1], we obtain

\[ \text{gap}(Κ^∗) ≤ \frac{\text{tr}(M')}{1 − σ_{max}^2 (A_{K_{lqr}})} = \frac{\text{tr}(M)}{1 − σ_{max}^2 (A_{K_{lqr}})} \]

where the last equality follows by the cyclic permutation property of trace and definition of \( A_{K}^∗ \).

**APPENDIX B**

**PROOF OF LEMMA 2**

First, we show that the following algebraic identities hold, which will be used in the proof of Lemma 2.

**Lemma 3:** Suppose that \( A \) and \( B \) are symmetric matrices such that \( A, A − B, \) and \( A + (n − 1)B \) are all invertible for some integer \( n \). Then, the following relations hold.

1. \( (A + nB)(A + (n − 1)B)^{-1}(A − B) = A − nB(A + (n − 1)B)^{-1}B \)
2. \( (A + nB)(A + (n − 1)B)^{-1}(A − B) = (A − B)(A + (n − 1)B)^{-1}B \)
3. \( (A + nB)(A − B)^{-1}B = B(A − B)^{-1}(A + nB) \)

**Proof:** These claims follow by the following algebraic manipulations: First

\[ (A + nB)(A + (n − 1)B)^{-1}(A − B) = (A + (n − 1)B) + B(A + (n − 1)B)^{-1}(A − B) \]
\[ = (I + B(A + (n − 1)B)^{-1})(A − B) \]
\[ = A − B + B(A + (n − 1)B)^{-1}A − B(A + (n − 1)B)^{-1}B \]
\[ = A + B(A + (n − 1)B)^{-1}A − B(A + (n − 1)B)^{-1}B \]
\[ = A + B(A + (n − 1)B)^{-1}(A − (A + (n − 1)B)) \]
\[ = B(A + (n − 1)B)^{-1}B \]
\[ = A − (n − 1)B(A + (n − 1)B)^{-1}B − B(A + (n − 1)B)^{-1}B \]
\[ = A − nB(A + (n − 1)B)^{-1}B \]

Second

\[ (A + nB)(A + (n − 1)B)^{-1}(A − B) = −n(A − B)(A + (n − 1)B)^{-1}(A − B) \]
\[ + (n + 1)A(A + (n − 1)B)^{-1}(A − B) \]
\[ = −n(A − B)(A + (n − 1)B)^{-1}(A − B) \]
\[ + (n + 1)(A − B)((I + (n − 1)B)^{-1}A − B)^{-1}(A − B) \]
\[ = −n(A − B)(A + (n − 1)B)^{-1}(A − B) \]
\[ + (n + 1)(A − B)((A − B)(I + (n − 1)A)^{-1}B)^{-1}(A − B) \]
\[ = −n(A − B)(A + (n − 1)B)^{-1}(A − B) \]
\[ + (n + 1)(A − B)(A + (n − 1)B)^{-1}B \]
\[ = (A − B)(A + (n − 1)B)^{-1}(A + nB) \]

Finally

\[ (A + nB)(A − B)^{-1}B = (A − B + (n + 1)B)(A − B)^{-1}B \]
\[ = (I + (n + 1)B(A − B)^{-1}B \]
\[ = B(I + (n + 1)(A − B)^{-1}B) \]
\[ = B(A − B)^{-1}(A + nB) \]

**Proof of Lemma 2**

1) We prove the claim by induction. First, note that both \( N_r \) and its principle submatrix \( A \) are invertible. For \( r = 2 \), by the Schur complement of \( N_2 \), we get
\[ \text{det}(N_2) = \text{det}(A) \text{det}(A − BA^{-1}B) \]
\[ = \text{det}(A) \text{det}(I − BA^{-1}B) \]
\[ = \text{det}(A) \text{det}(I + BA^{-1}B) \]
\[ = \text{det}(A) \text{det}(A + B) \]

Now, suppose that the claim holds for \( r = p \). Then, for \( r = p + 1 \), similarly by the Schur complement, we get
\[ \text{det}(N_{p+1}) = \text{det}(A) \text{det}(N_p − I I_{r} ⊗ BA^{-1}B) \]
\[ = \text{det}(A) \text{det}(A − B) \]
\[ \cdot \text{det}(A − BA^{-1}B + (p − 1)(B − BA^{-1}B)) \]
\[ = \text{det}(A) \text{det}(A − B) \]
\[ \cdot \text{det}(A − B + pBA^{-1}(A − B)) \]
\[ = \text{det}(A) \text{det}(A − B)^p \text{det}(I + pBA^{-1}) \]
\[ = \text{det}(A − B)^p \text{det}(A + pB) \]

where the second equality follows by applying the induction hypothesis to \( N_p − I I_{r} ⊗ BA^{-1}B \) and some algebraic manipulation. This completes the proof.

2) From item 1 of Lemma 2, we have
\[ \text{det}(N_r − λI_r ⊗ I) \]
\[ = \text{det}(A − λI − B)^{-1} \text{det}(A − λI + (r − 1)B) \]

implying that the spectrum of \( N_r \) coincides with that of \( A − B \) and \( A + (r − 1)B \) — modulo algebraic multiplicities. Hence, \( N_r \) results in \( A − B > 0 \) and
$A + (r - 1)B \succ 0$. Furthermore, $N_r \succ 0$ if and only if its principal submatrices are positive definite. Therefore, by applying the latter result to principal submatrices, we claim that $A + \ell B \succ 0$ for $\ell = 0, \ldots, r - 2$. Finally, from item 1 of Lemma 3, for $\ell = 1, \ldots, r - 1$, we have

$$A - \ell B (A + (\ell - 1)B)^{-1} B$$

which, by the first part of this claim, is invertible as a multiplication of invertible matrices.

3) Since $N_r$ and $A$ are invertible, the Schur complement $N_{r-1} - L_{r-1} A^{-1} L_{r-1}^\top$ is also invertible where $L_{r-1} = 1_{r-1} \otimes B$. We prove the claim by induction on $r$. For $r = 2$, by [63, Block matrix inverse formula (0.8.5.6)], we have

$$N_2^{-1} = \begin{bmatrix} H & -H^{-1}B A^{-1} \\ -A^{-1} B H^{-1} & A^{-1} + A^{-1} B H^{-1} B A^{-1} \end{bmatrix}$$

where $H = A - B A^{-1} B$ is the Schur complement of $A$. By [63, Woodbury inversion formula (0.7.4.1)], $H^{-1} = A^{-1} + A^{-1} B H^{-1} B A^{-1}$, establishing the recurrence of diagonal blocks. Also, $N_2$ is symmetric, so is $N_2^{-1}$ and thus establishing that $N_2 \in \text{PM}(2 \times n, \mathbb{R})$. Now, from item 1 in Lemma 3 with $n = 1$, we get

$$A^{-1} B H^{-1} = A^{-1} B (A - B)^{-1} A (A + B)^{-1}$$

$$= A^{-1} B (A + B)^{-1} (A - B)^{-1}$$

$$= A^{-1} B (I + A^{-1} B)^{-1} (A - B)^{-1}$$

$$= (I - (I + A^{-1} B)^{-1}) (A - B)^{-1} = G_2$$

where we also used $(A - B)^{-1} A (A + B)^{-1} = (A + B)^{-1} A (A - B)^{-1}$ derived from item 2 in Lemma 3. Hence

$$N_2^{-1} = I_2 \otimes \left( H^{-1} + H^{-1} B A^{-1} \right) - 1_{2 \times 2} \otimes \left( H^{-1} B A^{-1} \right)$$

Assume that the claim holds for $r = p$. To extend the result to $r = p + 1$, again by [63, Block matrix inverse formula (0.8.5.6)] and [63, Woodbury inversion formula (0.7.4.1)], we have

$$N_{p+1}^{-1} = \begin{bmatrix} A & L_p^\top \\ L_p & N_p \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} P^{-1} & -P^{-1} L_p^\top N_p^{-1} \\ -N_p^{-1} L_p^\top P^{-1} & (N_p - L_p A^{-1} L_p)^{-1} \end{bmatrix}$$

where $P = A - L_p^\top N_p^{-1} L_p$ and $L_p = 1_p \otimes B$. Let $N_p^{-1} = I_p \otimes (F_p + G_p) - 1_p \otimes B \otimes G_p$, where

$$F_p = \left( A - (p - 1) B (A + (p - 2) B)^{-1} B \right)^{-1}$$

$$G_p = (A + (p - 1) B)^{-1} B (A - B)^{-1}$$

where the inversions are valid from item 2 of Lemma 2. By simplification, we get $P = A - pB (F_p - (p - 1) G_p) B$, and from items 1 and 2 of Lemma 3 for $n = p - 1$, we have

$$F_p - (p - 1) G_p$$

$$= \left( A - (p - 1) B (A + (p - 2) B)^{-1} B \right)^{-1}$$

$$- (p - 1) (A + (p - 1) B)^{-1} B (A - B)^{-1}$$

$$= (A + (p - 1) B)^{-1} (A + (p - 2) B) (A - B)^{-1}$$

$$- (p - 1) (A + (p - 1) B)^{-1} B (A - B)^{-1}$$

$$= (A + (p - 1) B)^{-1}$$

where the first term on the right-hand side of the first equation undergoes items 1 and 2 in Lemma 3, consecutively. And, the last equality follows by a direct simplification. The latter established equality results in $P = A - pB (A + (p - 1) B)^{-1} B$. Next, considering the off-diagonal blocks of $N_{p+1}^{-1}$, with some simplification, each block of $P^{-1} L_p N_p^{-1}$ is equivalent to $P^{-1} B (F_p - (p - 1) G_p)$, and using the previous reasoning and Lemma 3, it can be simplified to

$$P^{-1} B (F_p - (p - 1) G_p)$$

$$= P^{-1} B (A + (p - 1) B)^{-1}$$

$$= (A + pB)^{-1} (A + (p - 1) B)$$

$$= (A - B)^{-1} B (A + (p - 1) B)^{-1}$$

$$= (A + pB)^{-1} (A - B)^{-1}.$$
REFERENCES

[1] S. Talebi, S. Alemzadeh, and M. Mesbahi, “Distributed model-free policy iteration for networks of homogeneous systems,” in Proc. IEEE 60th Conf. Decis. Control, pp. 6970–6975, 2021.

[2] D. A. Bini, B. Iannazzo, and B. Meini, Numerical Solution of Algebraic Riccati Equations. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 2011.

[3] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, “Distributed model predictive control,” IEEE Control Syst. Mag., vol. 22, no. 1, pp. 44–52, Feb. 2002.

[4] C. H. Papadimitriou and J. Tsitsiklis, “Intractable problems in control theory,” SIAM J. Control Optim., vol. 24, no. 4, pp. 639–654, 1986.

[5] D. M. Stipanović, G. Inalhan, R. Teo, and C. J. Tomlin, “Decentralized overlapping control of a formation of unmanned aerial vehicles,” Automatica, vol. 40, no. 8, pp. 1285–1296, 2004.

[6] F. Borrelli, T. Keviczky, G. J. Balas, G. Stewart, K. Fregene, and D. Godbole, “Hybrid decentralized control of large scale systems,” in Proc. Int. Workshop Hybrid Syst., Comput. Control, 2005, pp. 168–183.

[7] F. Borrelli and T. Keviczky, “Distributed LQR design for identical dynamically decoupled systems,” IEEE Trans. Autom. Control, vol. 53, no. 8, pp. 1901–1912, Sep. 2008.

[8] P. Deshpande, P. P. Menon, C. Edwards, and I. Postlethwaite, “A distributed convex cone law for the discrete optimal LQR cost for identical dynamically coupled linear systems,” in Proc. Amer. Control Conf., 2011, pp. 5342–5347.

[9] W. Wang, F. Zhang, and C. Han, “Distributed linear quadratic regulator control for discrete-time multi-agent systems,” IET Control Theory Appl., vol. 11, no. 14, pp. 2279–2287, 2017.

[10] P. Massioni and M. Verhaegen, “Distributed control for identical dynamically coupled systems: A decomposition approach,” IEEE Trans. Autom. Control, vol. 54, no. 1, pp. 124–135, Jan. 2009.

[11] J. Engwerda, LQ Dynamic Optimization and Differential Games. Hoboken, NJ, USA: Wiley, 2005.

[12] P. Ioannou, Decentralized adaptive control of interconnected systems, vol. AC-23, no. 2, pp. 108–128, Apr. 1978.

[13] J. Engwerda, Riccati Equations, vol. 40, no. 8, pp. 1285–1296, 2004.

[14] M. Rotkowitz and S. Lall, “A characterization of convex problems in decentralized control,” Found. Comput. Math., vol. 7, no. 6, pp. 743–775, 2007.

[15] H. A. Khargonekar, I. R. Petersen, and K. Zhou, Robust stabilization of uncertain linear systems: Quadratic stabilizability and \( H_{\infty} \) control theory,” IEEE Trans. Autom. Control, vol. 35, no. 3, pp. 356–361, Mar. 1990.

[16] K. Mårtensson and A. Rantzer, “Gradient methods for iterative distributed control,” IEEE Trans. Autom. Control, vol. 66, no. 11, pp. 5112–5127, Nov. 2021.

[17] S. Vasiht, “Data-guided estimation and tracking methods for unmanned aerial vehicles,” Ph.D. dissertation, Dept. Aeronaut. Astronaut., Univ. Washington, Seattle, WA, USA, 2019.

[18] P. Dorato, V. Cerone, and C. Abdallah, Linear-Quadratic Control: An Introduction. New York, NY, USA: Simon and Schuster, 1994.

[19] D. A. Bini, B. Iannazzo, and B. Meini, “Incremental dynamic programming for online adaptive optimal control,” IEEE Trans. Autom. Control, vol. 64, no. 11, pp. 4035–4049, Nov. 2019.

[20] S. Talebi, S. Alemzadeh, and M. Mesbahi, “Distributed model-free policy iteration for networks of homogeneous systems,” IEEE Trans. Autom. Control, vol. 66, no. 11, pp. 5493–5509, Nov. 2021.

[21] N. Motee and A. Jadbabaie, “Optimal control of spatially distributed systems,” IEEE Trans. Autom. Control, vol. 53, no. 11, pp. 2259–2264, Nov. 2008.

[22] Y. R. Stürz, A. Eichler, and R. S. Smith, “Distributed control design for heterogeneous interconnected systems,” IEEE Trans. Autom. Control, vol. 66, no. 11, pp. 5112–5127, Nov. 2021.

[23] N. Sandell, P. Varaiya, M. Athans, and M. Safonov, “Survey of decentralized control,” IEEE Trans. Autom. Control, vol. 28, no. 6, pp. 521–549, Dec. 1983.

[24] N. Sandell, P. Varaiya, M. Athans, and M. Safonov, “Survey of decentralized control,” IEEE Trans. Autom. Control, vol. 28, no. 6, pp. 521–549, Dec. 1983.

[25] P. Ioannou, Decentralized adaptive control of interconnected systems, vol. AC-23, no. 2, pp. 108–128, Apr. 1978.

[26] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, “A note on consistency of stabilizing feedback gain,” Int. J. Control, vol. 78, no. 16, pp. 1302–1313, 2005.

[27] P. Deshpande, P. Menon, C. Edwards, and I. Postlethwaite, “Sub-optimal distributed control law with \( H_{\infty} \) performance for identical dynamically coupled linear systems,” IET Control Theory Appl., vol. 6, no. 16, pp. 2509–2517, 2012.

[28] N. Motee and A. Jadbabaie, “Optimal control of spatially distributed systems,” IEEE Trans. Autom. Control, vol. 53, no. 7, pp. 1616–1629, Aug. 2008.

[29] A. Chapman, A. D. Gonzalez, M. Mesbahi, L. Dueñas-Osorio, and R. M. D’Souza, “Data-guided control: Clustering, graph products, and decentralized control,” in Proc. IEEE 58th Ann. Conf. Decis. Control, 2019, pp. 489–494.

[30] S. Alemzadeh and M. Mesbahi, “Influence models on layered uncertain networks: A guaranteed-cost design perspective,” in Proc. IEEE Conf. Decis. Control, 2018, pp. 5251–5256.

[31] Y. R. Stürz, A. Eichler, and R. S. Smith, “Distributed control design for heterogeneous interconnected systems,” IEEE Trans. Autom. Control, vol. 66, no. 11, pp. 5112–5127, Nov. 2021.

[32] S. Vasiht, “Data-guided estimation and tracking methods for unmanned aerial vehicles,” Ph.D. dissertation, Dept. Aeronaut. Astronaut., Univ. Washington, Seattle, WA, USA, 2019.
Y. Yu, S. Talebi, H. J. van Waarde, U. Topcu, M. Mesbahi, and B. Açıkmeşe, “On controllability and persistency of excitation in data-driven control: Extensions of Willems’ fundamental lemma,” in Proc. Conf. Decis. Control, 2021, pp. 6485–6490.

S. Talebi, S. Alemzadeh, N. Rahimi, and M. Mesbahi, “On regularizability and its application to online control of unstable LTI systems,” IEEE Trans. Autom. Control, vol. 67, no. 12, pp. 6413–6428, Dec. 2022.

Y. Jiang and Z.-P. Jiang, “Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics,” Automatica, vol. 48, no. 10, pp. 2699–2704, 2012.

F. L. Lewis and K. G. Vanvoudakis, “Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 41, no. 1, pp. 14–25, Feb. 2011.

P. Comon and G. H. Golub, “Tracking a few extreme singular values and vectors in signal processing,” Proc. IEEE, vol. 78, no. 8, pp. 1327–1343, Aug. 1990.

S. Haykin, Adaptive Filter Theory. Hoboken, NJ, USA: Prentice-Hall, 2002.

X. Chen and E. Hazan, “Black-box control for linear dynamical systems,” in Proc. 34th Conf. Learn. Theory, 2021, pp. 1114–1143.

Y. Abbasi-Yadkori and C. Szepesvári, “Regret bounds for the adaptive control of linear quadratic systems,” in Proc. 24th Annu. Conf. Learn. Theory, 2011, pp. 1–26.

N. Agarwal, B. Bullins, E. Hazan, S. Kakade, and K. Singh, “Online control with adversarial disturbances,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 111–119.

Y. S. Hung and A. MacFarlane, Multivariable Feedback: A Quasi-Classical Approach. New York, NY, USA: Springer-Verlag, 1982.

R. Bellman, “Dynamic programming,” Science, vol. 153, no. 3731, pp. 34–37, 1966.

W. Levine, T. Johnson, and M. Athans, “Optimal limited state variable feedback controllers for linear systems,” IEEE Trans. Autom. Control, vol. AC-16, no. 6, pp. 785–793, Dec. 1971.

R. A. Horn and C. R. Johnson, Matrix Analysis. Cambridge, U.K.: Cambridge Univ. Press, 2012.

P. Lancaster, “Explicit solutions of linear matrix equations,” SIAM Rev., vol. 12, no. 4, pp. 544–566, 1970.

M. A. Woodbury, “Inverting modified matrices,” Statist. Res. Group Memorandum Rep., vol. 42, no. 106, p. 336, 1950.

B. Bollobás, Modern Graph Theory, vol. 184. New York, York, USA: Springer, 2013.

G. C. Goodwin, S. F. Graebe, and M. E. Salgado, Control System Design. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

T. Mori, N. Fukuma, and M. Kuwahara, “On the discrete Lyapunov matrix equation,” IEEE Trans. Autom. Control, vol. AC-27, no. 2, pp. 463–464, Apr. 1982.

Shahriar Talebi (Member, IEEE) received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2014, and the M.Sc. degree in electrical engineering from the University of Central Florida, Orlando, FL, USA, in 2017. He is currently a Postdoctoral Fellow with the School of Engineering and Applied Sciences, Department of Statistics, Harvard University, Cambridge, MA, USA. His research interests include reinforcement learning, computational optimal control, networked dynamical systems, and game theory.

Dr. Talebi was the recipient of the 2022 Excellence in Teaching Award at the University of Washington (UW). He is also a recipient of a number of scholarships, including the William E. Boeing Endowed Fellowship, the Paul A. Carlsedt Endowment, and the Latvian Arctic Pilot-A. Vagners Memorial Scholarship (UW, 2018–2019), as well as the Frank Hubbard Engineering Scholarship (University of Central Florida, 2017).

Mehran Mesbahi (Fellow, IEEE) received the B.Sc. degree from the University of Southern California, Los Angeles, CA, USA, in 1996. From 1996 to 2000, he was a Member of the Guidance, Navigation, and Analysis Group, Jet Propulsion Laboratory, Pasadena, CA. From 2000 to 2002, he was an Assistant Professor of Aerospace Engineering and Mathematics with the University of Minnesota, Minneapolis, MN, USA. He is currently a Professor of Aeronautics and Astronautics and an Adjunct Professor of Electrical and Computer Engineering and Mathematics with the University of Washington (UW), Seattle, WA, USA, where he is also the Executive Director of the Joint Center for Aerospace Technology Innovation and a Member of the Washington State Academy of Sciences. His research interests include distributed and networked aerospace systems, systems and control theory, and learning.

Dr. Mesbahi was the recipient of the National Science Foundation CAREER Award, the NASA Space Act Award, the UW Distinguished Teaching Award, and the UW College of Engineering Innovator Award for Teaching.