Signal Processing 196 (2022) 108506

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Distributed linear-quadratic control with graph neural networks☆,☆☆

Fernando Gama[a,*], Somayeh Sojoudi[b]

[a] Department of Electrical and Computer Engineering, Rice University, Houston, TX, 77005 USA
[b] Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94709 USA

ARTICLE INFO

Article history:
Received 13 July 2021
Revised 29 December 2021
Accepted 11 February 2022
Available online 16 February 2022

Keywords:
Distributed control
Graph neural networks
Graph signal processing

ABSTRACT

Controlling network systems has become a problem of paramount importance. In this paper, we consider a distributed linear-quadratic problem and propose the use of graph neural networks (GNNs) to parametrize and design a distributed controller for network systems. GNNs exhibit many desirable properties, such as being naturally distributed and scalable. We cast the distributed linear-quadratic problem as a self-supervised learning problem, which is then used to train the GNN-based controllers. We also obtain sufficient conditions for the resulting closed-loop system to be input-state stable, and derive an upper bound on how much the trajectory deviates from the nominal value when the matrices that describe the system are not accurately known. We run extensive simulations to study the performance of GNN-based distributed controllers and show that they are computationally efficient and scalable.

1. Introduction

The use of linear models to describe dynamical systems has found widespread use in many areas of physics, mathematics, engineering and economics [2]. Linear systems are mathematically tractable and can thus be used to derive properties, draw insights, and improve on our ability to successfully control these systems. In particular, designing optimal controllers that can steer the system into a desired state while minimizing some given cost has become a problem of paramount importance [3].

Obtaining an optimal controller that minimizes a quadratic cost on the states and the actions, following a linear dynamic model, gives rise to the well-studied linear-quadratic control problem [4]. As it happens, the optimal linear-quadratic controller is *linear* and acts on the knowledge of the system state at a given time to produce the optimal control action for that time instant. Furthermore, when considering an infinite-time horizon for minimizing the quadratic cost, the resulting optimal controller is not only linear but also static, meaning that the same linear mapping is used between state and control action for all time instants.

Network systems are one particular class of dynamical systems that has become increasingly relevant. These systems are comprised of a set of interconnected components that are capable of exchanging information. They are further equipped with the ability to autonomously decide on an action to take based on the individual state of each component and the information relied through the communications with other neighboring components. The objective of controlling network systems is to coordinate the individual actions of the components so that they are conducive to the accomplishment of some global task [5].

The dynamics of some network systems can be effectively described by a linear model. Thus, if such systems are coupled with a quadratic cost, a corresponding linear-quadratic problem is obtained. As such, the optimal control actions are readily available. While the optimal controllers are linear, they require information from the components in the network irrespective of their interconnections. That is, to compute the optimal controller, an additional unit capable of accessing all components instantaneously is required. In the context of network systems, this is called a *centralized* approach.

Centralized controllers face limitations in terms of scalability and implementation. For increasingly large networks, the centralized unit requires more direct connections to all the components of the system. Similarly, the computational cost increases directly with the size of the network, since a single unit is responsible for computing the control actions of all the components. It is also less robust to changes in the network. A failed connection between the centralized unit and any of the components would render that component uncontrollable.

Network systems are characterized by the connections between components, which naturally impose a distributed structure on the flow of data. Fundamentally, it can be leveraged in the design of controllers. By doing so, one can overcome many of the

☆ This work is supported by grants from ONR, NSF and AFOSR.
☆☆ Partial results have appeared in [1].
* Corresponding author.
E-mail addresses: fgama@rice.edu (F. Gama), sojoudi@berkeley.edu (S. Sojoudi).

https://doi.org/10.1016/j.sigpro.2022.108506
0165-1684/

limitations of centralized controllers. Thus, we focus on leveraging the data structure to obtain *distributed* controllers. These are control actions that depend only on local information provided by components that share a connection and that can be computed separately by each component.

Imposing a distributed constraint on the linear-quadratic control problem renders it intractable in the most general case [6]. While there is a large class of distributed control problems that admit a convex formulation [7], many of them lead to complex solutions that do not scale with the size of the network [8]. An alternative approach is to adopt a linear parametrization of the controller and find a surrogate of the original problem that admits a scalable solution. The resulting controller is thus a sub-optimal linear distributed controller, and stability and robustness analyses are provided [9–11].

However, even in the context of linear network systems with a quadratic cost, the optimal distributed controller may not be linear [6]. In this paper, we thus adopt a nonlinear parametrization of the controller. More specifically, we focus on the use of graph neural networks (GNNs) [12]. GNNs consist of a cascade of blocks (commonly known as layers) each of which applies a bank of graph filters followed by a pointwise nonlinearity. GNNs exhibit several desirable properties in the context of distributed control. Most importantly, they are naturally local and distributed, meaning that by adopting a GNN as a mapping between states and actions, a distributed controller is automatically obtained. Furthermore, they are permutation equivariant and Lipschitz continuous to changes in the network [13]. These two properties allow them to scale up and transfer [14].

Distributed controllers leveraging neural network techniques can be found in [1,15–23]. These controllers typically use a distinct multi-layer perceptron (MLP) to parametrize the controller at each component [15–20] or rely on adaptive critic control [21,22]. Assigning a separate MLP to each component implies that the number of parameters to learn increases proportionally with the size of the network system, becoming increasingly harder to train, and thus this approach is not scalable. The use of GNNs imposes a weight-sharing scheme that avoids scalability problems. These are leveraged in [23] in the context of specific robotics problems. The distributed linear-quadratic problem using GNNs was investigated in our conference paper [1].

In this work, we focus on finding distributed controllers for the distributed linear-quadratic problem. Our main contributions are:

- **(C1)** We propose to parametrize the distributed controller with a GNN, obtaining a naturally distributed architecture that is capable of capturing nonlinear relationships between input and output, as it was initially investigated in our preliminary work [1].
- **(C2)** We obtain an improved sufficient condition for closed-loop input-state stability of the controller.
- **(C3)** We study the problem of systems whose linear description is not accurately known. We analyze how the stability of the system changes and obtain an upper bound on the deviation of the trajectory from its nominal value.
- **(C4)** We present new simulations that provide better insight into GNN-based controllers for a distributed LQR problem.

The remainder of this paper is organized as follows. We formulate the linear-quadratic problem in Section 2 and postulate the use of graph neural networks in Section 3 as a practically useful nonlinear parametrization of the unknown distributed controller. We cast the distributed linear-quadratic problem as a self-supervised learning problem, which can be efficiently solved by traditional machine learning techniques. To study the effect of adopting a GNN-based controller on the entire dynamical system, we obtain a sufficient condition for the resulting closed-loop system to be input-state stable and derive an upper bound on the trajectory deviation from its nominal value when the system matrices are unknown and only estimates are available. We include numerical simulations in Section 5 to investigate the performance of GNN-based distributed controllers and their dependence on design hyperparameters, as well as their scalability. Conclusions are drawn in Section 6. Proofs are provided in the appendix.

2. The linear-quadratic problem

The linear-quadratic problem is one of the fundamental problems in optimal control theory [3]. Consider a system described by a state vector $\mathbf{x}(t) \in \mathbb{R}^F$ and controlled by an action $\mathbf{u}(t) \in \mathbb{R}^G$ at time $t \in \{0, 1, 2, \ldots\}$. The system evolves following a linear dynamic

$$\mathbf{x}(t+1) = \bar{\mathbf{A}}\mathbf{x}(t) + \bar{\mathbf{B}}\mathbf{u}(t) \tag{1}$$

determined by $\bar{\mathbf{A}} \in \mathbb{R}^{F \times F}$ called the *system matrix* and $\bar{\mathbf{B}} \in \mathbb{R}^{F \times G}$ called the *control matrix*. These two matrices are considered to be known and given in the problem formulation. The objective is to drive the system towards a desired, target state value. To this end, a *controller* $\Phi : \mathbb{R}^F \to \mathbb{R}^G$ that maps the current state of the system $\mathbf{x}(t)$ into an appropriate action $\mathbf{u}(t) = \Phi(\mathbf{x}(t))$ is typically designed. In optimal control, it is desirable to find a controller that minimizes a given cost. In particular, the focus here is on the quadratic cost given by

$$\mathrm{J}\Big(\{\mathbf{x}(t)\}, \{\mathbf{u}(t)\}\Big) = \sum_{t=0}^{\infty} \Big(\mathbf{x}(t)^{\mathsf{T}} \bar{\mathbf{Q}} \mathbf{x}(t) + \mathbf{u}(t)^{\mathsf{T}} \bar{\mathbf{R}} \mathbf{u}(t)\Big) \tag{2}$$

for two known matrices $\bar{\mathbf{Q}} \in \mathbb{R}^{F \times F}$ and $\bar{\mathbf{R}} \in \mathbb{R}^{G \times G}$ such that $\bar{\mathbf{Q}} \succeq 0$ and $\bar{\mathbf{R}} \succ 0$, given in the problem formulation.

The linear-quadratic problem can be formulated as

$$\min_{\Phi \in \mathbf{\Phi}} \mathrm{J}\Big(\{\mathbf{x}(t)\}, \{\mathbf{u}(t)\}\Big) \tag{3a}$$

$$\text{s.t. } \mathbf{x}(t+1) = \bar{\mathbf{A}}\mathbf{x}(t) + \bar{\mathbf{B}}\mathbf{u}(t), \qquad \forall t \in \{0, 1, \ldots\} \tag{3b}$$

$$\mathbf{u}(t) = \Phi(\mathbf{x}(t)), \qquad \forall t \in \{0, 1, \ldots\} \tag{3c}$$

where $\mathbf{\Phi}$ is the space of all functions $\Phi : \mathbb{R}^F \to \mathbb{R}^G$, see [3]. The objective function (3a) is the quadratic cost (2), the constraint (3b) imposes the linear dynamics of the system (1) and the constraint (3c) forces the solution to be a function $\Phi : \mathbb{R}^F \to \mathbb{R}^G$. The optimal controller obtained from solving (3) is formally known as a *linear-quadratic regulator* (LQR) and is given by

$$\mathbf{u}^\star(t) = \Phi^\star\big(\mathbf{x}(t)\big) = \mathbf{K}^\star \mathbf{x}(t), \tag{4}$$

with $\mathbf{K}^\star \in \mathbb{R}^{F \times G}$ being a linear operator that depends on the matrices that describe the problem, namely $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{Q}}, \bar{\mathbf{R}}$, and can be readily computed [3, Sec. 2.4]. Notably, the LQR is a linear controller [3, eq. (2.4–8)].

A network system can be conveniently described by means of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The node v_i represents the i^{th} component of the system, while the existence of the edge $(v_i, v_j) \in \mathcal{E}$ implies that nodes v_i and v_j are interconnected and capable of exchanging information. In a network system, each node is described by a state $\mathbf{x}_i(t) \in \mathbb{R}^F$ and is capable of autonomously taking an action $\mathbf{u}_i(t) \in \mathbb{R}^G$ at time t. The states and actions of all nodes are collected in two matrices $\mathbf{X}(t) \in \mathbb{R}^{N \times F}$ and $\mathbf{U}(t) \in \mathbb{R}^{N \times G}$, respectively, where each row corresponds to the state or action of each agent.

Similar to (1), consider a network system with linear dynamics modeled as

$$\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t)\bar{\mathbf{A}} + \mathbf{B}\mathbf{U}(t)\bar{\mathbf{B}}, \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is called the *network system matrix* and $\mathbf{B} \in \mathbb{R}^{N \times N}$ the *network control matrix*. The linear system in (5) is an extension of (1) tailored to handle network data. In particular, it considers that each node v_i is described by an F-dimensional state $\mathbf{x}_i(t)$, collected in the rows of the matrix $\mathbf{X}(t)$. It also decouples the impact that the network topology has on the evolution of the system (through matrices $\mathbf{A}$ and $\mathbf{B}$) from the impact that the individual states have (through $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$). To see this, note that matrices $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{B} \in \mathbb{R}^{N \times N}$ act as linear combinations of state values across different nodes, and as such, these combinations are typically restricted to follow the interconnection of the components (although, technically, they need not be). It is thus noted that while the matrix $\mathbf{A}$ need not be the adjacency matrix of the graph, it is usually a function of it –for example, both matrices may share the same eigenvectors. The matrices $\bar{\mathbf{A}} \in \mathbb{R}^{F \times F}$ and $\bar{\mathbf{B}} \in \mathbb{R}^{G \times F}$ determine the evolution of the values of the state at each individual node and, while they can be arbitrary, they force all individual state nodes to follow the same evolution. Finally, it is noted that, while a more general linear description can be obtained by adopting a network state of dimension NF and using (1), doing so obscures the effect of the topology of the network on the evolution of the system. Thus, (5) is adopted from now on for mathematical simplicity –and it is observed that all the results derived from here onward hold for (1) as well.

To pose the linear-quadratic problem for a network system, the following quadratic cost as a counterpart of (2) is adopted:

$$\mathrm{J}\Big(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}\Big) = \sum_{t=0}^{\infty} \Big(\|\mathbf{X}(t)\bar{\mathbf{Q}}^{1/2}\|_F^2 + \|\mathbf{U}(t)\bar{\mathbf{R}}^{1/2}\|_F^2 \Big), \tag{6}$$

where $\bar{\mathbf{Q}} \in \mathbb{R}^{F \times F}$ and $\bar{\mathbf{R}} \in \mathbb{R}^{G \times G}$ are two given positive definite matrices, and where $\|\cdot\|_F$ denotes the Frobenius matrix norm. The linear-quadratic control problem for a network system can then be posed in the form of (3), by replacing the cost (3a) with (6), the linear dynamics (3b) with (5), and the controller (3c) with one such that $\Phi : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times G}$.

The controller solving the linear-quadratic problem for a network system is also linear. However, in order to compute the optimal control action, the system needs to access the state of arbitrary components of the system, beyond those directly connected. This constitutes a *centralized* controller. In what follows, the focus is on finding a *distributed* controller.

A distributed controller, which is denoted by $\Phi(\mathbf{X}(t); \mathcal{G})$ to emphasize its dependence on the topology of the network system $\mathcal{G}$, should satisfy the properties that the control actions $\mathbf{U}(t)$ rely only on local information provided by other components that share a direct connection, and that they can be computed separately at each component. The use of a distributed controller overcomes some of the issues that arise when considering a centralized one. Namely, they are expected to scale better, since they do not require a single unit to compute the actions of all components in the system, and they are easy to implement since they do not demand an infrastructure capable of connecting all components to the single centralized unit.

The distributed linear-quadratic problem can be written as

$$\min_{\Phi \in \mathbf{\Phi}_{\mathcal{G}}} \mathrm{J}\Big(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}\Big) \tag{7a}$$

$$\text{s.t.}\ \mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t)\bar{\mathbf{A}} + \mathbf{B}\mathbf{U}(t)\bar{\mathbf{B}}, \qquad \forall t \in \{0, 1, \ldots\} \tag{7b}$$

$$\mathbf{U}(t) = \Phi\Big(\mathbf{X}(t); \mathcal{G}\Big), \qquad \forall t \in \{0, 1, \ldots\}, \tag{7c}$$

where $\mathbf{\Phi}_{\mathcal{G}}$ is the space of all functions $\Phi(\cdot; \mathcal{G}) : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times G}$ that can be computed in a distributed manner (i.e. relying only on local information and computed separately at each component). It is noted that the constraint (7c) further restricts the feasible set, and as such, the optimal value $J^\star_{\mathcal{G}}$ of solving (7) is lower bounded by the optimal value $J^\star$ incurred when using the optimal centralized controller, i.e. $J^\star_{\mathcal{G}} \geq J^\star$.

Solving problem (7) requires solving an optimization problem over the space of functions $\mathbf{\Phi}_{\mathcal{G}}$. This is mathematically intractable in the general case, and requires specific approaches involving variational methods, dynamic programming or kernel-based functions [24]. While there is a large class of distributed control problems that admit a convex formulation [7], many of them lead to complex solutions that do not scale with the size of the network [8].

Considering the inherent complexities of functional optimization, a popular approach is to adopt a specific model for the mapping Φ, leading to a parametric family of controllers. Inspired by the linear nature of the optimal centralized solution and its mathematical tractability, a distributed linear parametrization was adopted in [9,11]. Many properties of this parametric family of controllers have been studied, including stability, robustness and (sub)optimality [9,11].

However, it is known that the linear system (5) may have a nonlinear optimal controller if we force a distributed nature on its solution [6]. This suggests that it would be more convenient to work with nonlinear parametrizations, rather than linear ones. In particular, this work focuses on graph neural networks (GNNs) [12]. These are nonlinear mappings that exhibit several desirable properties. Fundamentally, they are naturally computed in a distributed manner relying only on local information provided by directly connected components. This implies that any controller that is parametrized by means of a GNN respects the distributed nature of the system (as given by the graph $\mathcal{G}$), naturally incorporating the distributed constraint (7c) into the chosen parametrization.

3. Graph neural networks

Finding the optimal distributed controller by solving problem (7) is intractable in its most general case. This is due to the constraint (7c) that the solution satisfies a distributed computation. In what follows, a parametric family of distributed controllers is adopted. More concretely, inspired by the fact that the optimal controller is usually nonlinear, GNN-based controllers are considered. The basics of graph signal processing are introduced in Section 3.1, which allows for the definition of GNNs in Section 3.2. A discussion on how to cast the resulting finite-dimensional optimal control problem as an unsupervised learning problem follows in Section 3.3.

3.1. Graph signal processing

Graph signal processing (GSP) is a framework tailored to describe, analyze, and understand distributed problems [25]. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that describes the structure of the data under study, a *graph signal* $\mathsf{x} : \mathcal{V} \to \mathbb{R}$ is defined as a mapping from the nodes of the graph to a real number. By imposing an arbitrary order on the nodes, this graph signal can be conveniently described as a vector $\mathbf{x} \in \mathbb{R}^N$ whose i^{th} element corresponds to the signal value associated to node v_i, denoted as $[\mathbf{x}]_i = \mathsf{x}(v_i) = x_i \in \mathbb{R}$. Note that $[\cdot]_i$ ($[\cdot]_{ij}$) denotes the value of the i^{th} ($(i,j)^{\text{th}}$) entry of a vector (matrix). To be able to use the concept of graph signals to describe the state $\mathbf{X}(t) \in \mathbb{R}^{N \times F}$ and the control action $\mathbf{U}(t) \in \mathbb{R}^{N \times G}$ in a network system, an extension to vector-valued mappings is needed. Define the *vector-valued graph signal* as $\mathsf{X} : \mathcal{V} \to \mathbb{R}^F$, where $\mathsf{X}(v_i) = \mathbf{x}_i \in \mathbb{R}^F$. It can then be described by means of a matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$, where each row corresponds to the signal value at each

node

$$\mathbf{X} = \begin{bmatrix} \mathsf{X}^{\mathsf{T}}(v_1) \\ \vdots \\ \mathsf{X}^{\mathsf{T}}(v_N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_N^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^1 & \cdots & \mathbf{x}^F \end{bmatrix}. \tag{8}$$

In this equation, the vector-valued graph signal X is viewed as a collection of F traditional scalar-valued graph signals $\{\mathsf{x}^f\}_{f=1}^F$, placed in the columns of the matrix. Observe that x stands for the graph signal as a function, x_i stands for the scalar value adopted by node v_i, $\mathbf{x}$ for the vector collecting all these values; likewise, X stands for the vector-valued graph signal and $\mathbf{X}$ for the matrix collecting all the states at all nodes. All these quantities are related to the graph signal that is used to describe the state of the system. The *size of the vector-valued graph signal* is defined as

$$\|\mathbf{X}\| = \|\mathbf{X}\|_{2,1} = \sum_{f=1}^{F} \|\mathbf{x}^f\|_2. \tag{9}$$

The $L_{2,1}$ norm for matrices (9) is chosen as the size of the graph signal norm for both its robustness and its mathematical tractability. Note that if $F = 1$, then $\|\mathbf{X}\| = \|\mathbf{x}\|_2$ as expected. Finally, note that, in what follows, the term "graph signal" is used indistinctly to refer to either vector-valued or scalar-valued ones. Note that the trajectories of system states $\{\mathbf{X}(t)\}$ and control actions $\{\mathbf{U}(t)\}$ can each be modeled as a sequence of graph signals, indexed by the time parameter t —also known as graph processes [26].

Describing a graph signal in terms of a matrix is convenient because it allows for easy mathematical manipulation. However, this causes the loss of the information related to the underlying graph support. To recover this information, the graph is described in terms of a *support matrix* $\mathbf{S} \in \mathbb{R}^{N \times N}$ that respects the sparsity of the graph, i.e. $[\mathbf{S}]_{ij} = s_{ij}$ can be nonzero if and only if $i = j$ or $(v_j, v_i) \in \mathcal{E}$. Any matrix that satisfies this condition can be used as a support matrix and thus it is a design choice. Typical choices include the adjacency matrix, the Laplacian matrix, the Markov matrix, and their normalized counterparts [25]. A linear mapping $\mathsf{S} : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times F}$ between graph signals that relates the input to the underlying graph support $\mathbf{Y} = \mathsf{S}(\mathbf{X}) = \mathbf{SX}$ can be defined, such that the $(i, f)^{\text{th}}$ entry y_i^f of the matrix $\mathbf{Y}$ (the value of the f^{th} scalar graph signal at node v_i) is computed as

$$y_i^f = [\mathbf{Y}]_{if} = [\mathbf{SX}]_{if} = \sum_{j=1}^{N} [\mathbf{S}]_{ij} [\mathbf{X}]_{jf} = \sum_{j: v_j \in \mathcal{N}_i \cup \{v_i\}} s_{ij} x_j^f, \tag{10}$$

where $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$ is the set of nodes that share an edge with v_i and $[\mathbf{X}]_{jf} = [\mathbf{x}_j]_f = [\mathbf{x}^f]_j = x_j^f$, see (8). The last equality in (10) holds because of the sparsity pattern of the support matrix $\mathbf{S}$ and implies that the computation of the value of the output graph signal $\mathbf{Y}$ at node v_i only requires information relied by its neighbors. In this respect, one can then think of the pair $(\mathbf{X}, \mathbf{S})$ as the complete graph data containing all the relevant information; however, only $\mathbf{X}$ is regarded as the *actionable* variable (the signal), while the support $\mathbf{S}$ is considered given and fixed and is determined by the physical constraints of the network.

The support matrix $\mathbf{S}$ can be thought of as a linear mapping between graph signals that effectively relates the input to the underlying graph support. As such, the operation $\mathbf{SX}$ becomes the basic building block of graph signal processing [25]. A finite-impulse response (FIR) graph filter $\mathsf{H} : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times G}$ is a linear operation between two graph signals, defined as a polynomial on $\mathbf{S}$

$$\mathbf{Y} = \mathsf{H}(\mathbf{X}; \mathbf{S}, \mathcal{H}) = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X} \mathbf{H}_k, \tag{11}$$

where $\mathcal{H} = \{\mathbf{H}_k \in \mathbb{R}^{F \times G}, k = 0, \ldots, K\}$ is the set of filter taps $\mathbf{H}_k$ that characterize the filter response. The filter (11) is linear in the input $\mathbf{X}$ and is capable of mapping between vector-valued graph signals of different dimensions (but defined on the same graph given by $\mathbf{S}$).

The graph filter is a naturally distributed operation, meaning that the output of filtering in (11) can be computed separately by each node relying only on information provided by one-hop neighbors. To understand this, note that multiplications to the left of $\mathbf{X}$ carry out a linear combination of signal values across different nodes, and thus this matrix needs to respect the sparsity of the graph so that only values at neighboring nodes are combined. This is the case for $\mathbf{S}^k = \mathbf{S}^{k-1}\mathbf{S}$ which amounts to communicating k times with the one-hop neighbors. Therefore, an FIR graph filter is a distributed linear operation since it requires only K communication exchanges with one-hop neighbors. Multiplications to the right of $\mathbf{X}$, on the other hand, are linear combinations of signal values located at the same node, and can thus be arbitrary. In particular, (11) imposes a weight-sharing scheme, where the signal values at all nodes are combined in the same way. Finally, note that (11) is a compact notation for denoting the graph filtering operation but, in practice, the nodes do not need access to the full matrix $\mathbf{S}$. They only need access to the entries corresponding to their one-hop neighbors in order to compute the proper linear combination indicated in (10). Thus, in practice, the nodes need not know the entire graph topology.

The FIR graph filter (11) can be understood as a bank of FG filters acting on scalar-valued graph signals, see [12,27]. It can thus be characterized by its frequency response given by the collection of univariate polynomials

$$\left\{ h_{fg}(\lambda) = \sum_{k=0}^{K} [\mathbf{H}_k]_{fg} \lambda^k : \lambda \in [\lambda_l, \lambda_h] \;\; f = 1, \ldots, F \;\; g = 1, \ldots, G \right\}. \tag{12}$$

The values of λ_l and λ_h are determined by the specific problem under study, and are typically set to be the minimum and maximum eigenvalues of the given $\mathbf{S}$. However, they may be different if the problem requires the filters to be able to act on more than one graph, see Section 4.3. In that case, it may be convenient to select the interval so that it contains all the eigenvalues of all the support matrices under consideration.

The characterization of the filter in terms of the frequency response (12) allows for the definition of the *size of the graph filter* as

$$C_{\mathsf{H}} = \|\mathbf{C}_{\mathsf{H}}\|_{\infty} \quad \text{with} \quad \mathbf{C}_{\mathsf{H}} \in \mathbb{R}^{F \times G} : [\mathbf{C}_{\mathsf{H}}]_{fg} = \max_{\lambda \in [\lambda_l, \lambda_h]} |h_{fg}(\lambda)|. \tag{13}$$

In what follows, the focus is further set on a particular class of graph filters, known as *Lipschitz* filters. The graph filter (11) is said to be a Lipschitz filter if its frequency response (12) satisfies that

$$|h_{fg}(\lambda_1) - h_{fg}(\lambda_2)| \leq \gamma_{fg} |\lambda_1 - \lambda_2|, \quad \forall \lambda_1, \lambda_2 \in [\lambda_l, \lambda_h], \tag{14}$$

for some constant $\gamma_{fg} > 0$, for all $f \in \{1, \ldots, F\}$ and $G \in \{1, \ldots, G\}$. The Lipschitz constant Γ_{H} of the filter is computed as

$$\Gamma_{\mathsf{H}} = \|\mathbf{\Gamma}_{\mathsf{H}}\|_{\infty} \quad \text{with} \quad \mathbf{\Gamma}_{\mathsf{H}} \in \mathbb{R}^{F \times G} : [\mathbf{\Gamma}_{\mathsf{H}}]_{fg} = \gamma_{fg}, \tag{15}$$

which is the infinity norm $\|\mathbf{\Gamma}_{\mathsf{H}}\|_{\infty}$ for a matrix $\mathbf{\Gamma}_{\mathsf{H}} \in \mathbb{R}^{F \times G}$ containing the corresponding Lipschitz constants of each individual filter (i.e. the maximum absolute row sum of the matrix).

3.2. Graph neural networks

Graph filters are distributed, linear operations and, as such, are only capable of capturing linear relationships between input and output. However, the objective of this work is to learn nonlinear distributed controllers. Arguably, the most straightforward way of converting a graph filter into a nonlinear processing unit without

affecting its distributed nature is to include a pointwise nonlinearity

$$\mathbf{Y} = \sigma\Big(\mathsf{H}(\mathbf{X}; \mathbf{S}, \mathcal{H})\Big), \tag{16}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinearity applied pointwise to the entries of the graph signal obtained from applying the graph filter, i.e. $[\sigma(\mathbf{X})]_{if} = \sigma([\mathbf{X}]_{if})$. The operation (16) is known as a *graph perceptron* [12] and, since the nonlinearity $\sigma(\cdot)$ is applied pointwise to the entries of the graph signal, it retains the distributed nature of the graph filter.

The graph perceptron (16) is a nonlinear processing unit, but it has a limited representation power. To overcome this, a *graph convolutional neural network* $\Phi(\cdot; \mathbf{S}, \mathcal{H}) : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times G}$ is defined as a cascade of L graph perceptron units

$$\mathbf{X}_\ell = \sigma\Big(\mathsf{H}_\ell(\mathbf{X}_{\ell-1}; \mathbf{S}, \mathcal{H}_\ell)\Big), \tag{17a}$$

$$\Phi(\mathbf{X}; \mathbf{S}, \mathcal{H}) = \mathbf{X}_L, \tag{17b}$$

with $\mathcal{H} = \cup_{\ell=1}^{L} \mathcal{H}_\ell$. The input to the first layer is the graph signal $\mathbf{X}_0 = \mathbf{X}$ and the output is collected at the last layer. The space of all possible representations obtained by using a GNN is characterized by the set of filter taps $\mathcal{H}$, which contains the filter coefficients $\mathcal{H}_\ell = \{\mathbf{H}_{\ell k} \in \mathbb{R}^{F_{\ell-1} \times F_\ell} \quad k = 0, 1, \ldots, K_\ell\}$ at each layer $\ell \in \{1, \ldots, L\}$. Note that $F_0 = F$ and $F_L = G$. The nonlinear function $\sigma(\cdot)$, the number of layers L, the dimension of the graph signals at each layer F_ℓ and the number of filter taps at each layer K_ℓ are design choices and are typically referred to as *hyperparameters* [28].

3.3. Self-supervised learning

The linear graph filter (11) and the nonlinear GNN (17) have been introduced as naturally distributed parametrizations. By choosing to adopt one of these models for the to-be-learned controller, the focus is immediately set on a distributed mapping between the state and the action, turning the functional optimization problem (7) into the finite-dimensional optimization

$$\min_{\mathcal{H}} \mathsf{J}\Big(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}\Big) \tag{18a}$$

$$\text{s. t.} \; \mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t)\bar{\mathbf{A}} + \mathbf{B}\mathbf{U}(t)\bar{\mathbf{B}}, \tag{18b}$$

$$\mathbf{U}(t) = \Phi\Big(\mathbf{X}(t); \mathbf{S}, \mathcal{H}\Big). \tag{18c}$$

The constraint (18c) replaces a generic distributed controller $\Phi(\mathbf{X}(t); \mathcal{G})$ in (7c) with a controller that admits a parametrization based on either a graph filter or a GNN. The resulting controller $\Phi(\mathbf{X}(t); \mathbf{S}, \mathcal{H}^\star)$ with filter coefficients $\mathcal{H}^\star$ that solves (18) naturally satisfies the distributed constraint.

Problem (18) is nonconvex when adopting a GNN-based controller (18c). Thus, to approximately solve this problem, the empirical risk minimization (ERM) approach that is typical in learning theory [29] is leveraged. To do this, a *training set* $\mathcal{T} = \{\mathbf{X}_{1,0}, \ldots, \mathbf{X}_{|\mathcal{T}|,0}\}$ containing $|\mathcal{T}|$ samples $\mathbf{X}_{p,0}$ drawn independently from some distribution p is considered to be the different random initializations of the system. Then, the ERM problem is given by

$$\min_{\mathcal{H}} \sum_{p=1}^{|\mathcal{T}|} \mathsf{J}\Big(\{\mathbf{X}_p(t)\}, \{\mathbf{U}_p(t)\}\Big) \tag{19a}$$

$$\text{s. t.} \;\; \mathbf{X}_p(t+1) = \mathbf{A}\mathbf{X}_p(t)\bar{\mathbf{A}} + \mathbf{B}\mathbf{U}_p(t)\bar{\mathbf{B}}, \tag{19b}$$

$$\mathbf{U}_p(t) = \Phi\Big(\mathbf{X}_p(t); \mathbf{S}, \mathcal{H}\Big), \tag{19c}$$

$$\mathbf{X}_p(0) = \mathbf{X}_{p,0}. \tag{19d}$$

Problem (19) can be solved by means of an algorithm based on stochastic gradient descent [30], efficiently computing the gradient of $\mathsf{J}(\cdot, \cdot)$ with respect to the parameter $\mathcal{H}$ by means of the back-propagation algorithm [31]. To estimate the performance of the learned controllers –i.e. those obtained by solving (19)– a new set of initial states is generated, called the test set, and the average quadratic cost (6) is computed on the resulting trajectories. In essence, the optimization problem (18) is transformed into a self-supervised ERM problem (19) that is solved through simulated data.

It is observed that, during the training phase, the optimization problem (19) has to be solved in a centralized manner due to the weight-sharing scheme imposed by the FIR graph filters (recall that this weight-sharing scheme is necessary for scalability, keeping the number of learnable parameters independent of the size of the graph). However, this training phase can be carried out offline, prior to online execution. Once the GNN-based controllers are learned and the training phase is finished, they can be deployed in an entirely distributed manner for testing in the online phase. It is noted that there exist distributed optimization algorithms that leverage consensus to arrive to the optimal set of filter taps $\mathcal{H}$ [32]. These techniques, however, are outside the scope of the present work and will be left as future research directions.

4. Properties of GNN controllers

GNNs have many suitable properties that make them appropriate choices for learning distributed controllers. As standalone processing units, they are naturally distributed architectures and have the properties of permutation equivariance and Lipschitz continuity to changes in the underlying graph support. As part of a linear dynamical system, GNN-based controllers can also be shown to stabilize the system. Furthermore, the deviation in the nominal trajectory due to unknown system matrices can be mitigated with properly learned filters. These properties, which are studied in this section, hold for any GNN controller of the form (17) that satisfy the corresponding hypotheses.

4.1. GNN Properties

The main motivation for choosing GNNs as parametrizations for the controller is that they are naturally distributed architectures. GNNs are built by using graph filters and pointwise nonlinearities. Graph filters are distributed operations, as discussed after (11). The pointwise nonlinearity does not affect this, and thus GNNs are also distributed. It is noted that asynchronous implementations of graph filtering are possible [33]. Additionally, GNNs are capable of learning nonlinear controllers, which is a key feature in the context of distributed control, as it is expected that optimal distributed controllers to be nonlinear [6].

GNNs exhibit the property of permutation equivariance, [13, Prop. 2], which means that a reordering of the nodes does not affect the output, since it will be correspondingly reordered. This further implies that the GNNs are capable of leveraging any existing symmetries in the underlying graph topology to improve training. More specifically, learning how to process a given signal from the training set means that the GNN learns how to process the same signal anywhere in the graph with the same neighborhood topology. In a manner akin to the data augmentation that happens naturally by the choice of the convolution operation in regular convolutional neural networks (CNNs), permutation equivariance shows

precisely one way in which the GNN exploits the data structure to improve training and generalization.

GNNs are also Lipschitz continuous to changes in the underlying graph [13, Thm. 4]. This means that, if the underlying graph support is perturbed, the output of the GNN changes linearly with the size of perturbation. This implies that a GNN trained on one graph but tested on another one will still work well as long as both graphs are similar, see [14]. It also implies that if the graph is not known exactly but has to be estimated, then the GNN can still be trained as long as the graph support estimate is good enough. Additionally, it indicates that GNNs are suitable for time-varying scenarios where the changes to the graph support are slow [23].

4.2. Closed-Loop stability

GNNs have many suitable properties for learning distributed controllers. However, this does not necessarily guarantee that they are a good choice for a control system. In what follows, properties relating to GNN-based controllers within a linear dynamical system are studied.

A network system with the linear dynamics (5) is characterized by the set of matrices $\mathcal{D} = \{\mathbf{S}, \mathbf{A}, \bar{\mathbf{A}}, \mathbf{B}, \bar{\mathbf{B}}\}$, where $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the graph support matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\bar{\mathbf{A}} \in \mathbb{R}^{F \times F}$ are the system matrices, and $\mathbf{B} \in \mathbb{R}^{N \times N}$ and $\bar{\mathbf{B}} \in \mathbb{R}^{G \times F}$ are the control matrices. The trajectory of the system $\{\mathbf{X}(t)\}$ depends on these matrices. GNNs are capable of stabilizing the closed-loop dynamics of a distributed linear system $\mathcal{D}$. More specifically, drawing from [34], the notion of input-state stability is defined as follows.

Definition 1 (Input-state stability)**.** Consider a linear dynamical system as in (5) controlled by $\mathbf{U}(t) = \Phi(\mathbf{X}(t)) + \mathbf{E}(t)$ where $\mathbf{E}(t)$ is a disturbance term or exploratory signal. The system is input-state stable if, for all sequences $\{\mathbf{X}(t)\}$ and $\{\mathbf{E}(t)\}$ such that $\sum_{t=0}^{\infty} \|\mathbf{X}(t)\| < \infty$ and $\sum_{t=0}^{\infty} \|\mathbf{E}(t)\| < \infty$, there exist constants $\beta_0, \beta_1 \geq 0$ such that

$$\sum_{t=0}^{\infty} \|\mathbf{X}(t)\| \leq \beta_0 + \beta_1 \sum_{t=0}^{\infty} \|\mathbf{E}(t)\|. \tag{20}$$

This definition of input-state stability is widely used [34]. Given a trained GNN-based controller, a sufficient condition for the resulting system to be stable can be determined.

Theorem 1 (Sufficient condition for input-state stability)**.** *Consider a distributed linear system $\mathcal{D}$. Assume that the system is controlled with a GNN (17) consisting of L layers of filters $\mathrm{H}_\ell(\cdot; \mathbf{S}, \mathcal{H})$ with F_ℓ features and K_ℓ taps each. Let the nonlinearity $\sigma(\cdot)$ be such that $|\sigma(x)| \leq |x|$. Then, the closed-loop system is input-state stable if it holds that*

$$\xi(\mathcal{D}, \mathcal{H}) < 1, \tag{21}$$

where

$$\xi(\mathcal{D}, \mathcal{H}) = \|\mathbf{A}\|_2 \|\bar{\mathbf{A}}\|_\infty + C_\Phi \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty \tag{22}$$

is the stability constant, with $C_\Phi = \prod_{\ell=1}^{L} C_{\mathrm{H}_\ell}$ for C_{H_ℓ} the size of the ℓ^{th} filter, see (13).

Proof. See Appendix B. □

Theorem 1 is a sufficient condition for the closed-loop system to be input-state stable. The learned filters affect the constant C_Φ such that the smaller the filters C_{H_ℓ} the smaller C_Φ and thus ξ. Therefore, a penalty on the size of the filters, see (13), can be added to the objective function of (19) to obtain GNNs with a controlled value of C_Φ and therefore with a smaller stability constant ξ. The condition on the nonlinearity is mild and is satisfied by the most popular nonlinearities (ReLU, tanh, sigmoid, etc.). It is observed that the sufficient condition requires $\|\mathbf{A}\|_2 \|\bar{\mathbf{A}}\|_\infty < 1$, which implies that the system is open-loop stable. In many physical systems such as power networks, it is possible to design stabilizing controllers. This implies that once the system has been stabilized a GNN-based controller can then be learned to minimize the quadratic cost.

4.3. Trajectory deviation

It often happens that one does not have direct access to the matrices $\mathcal{D}$ that characterize the distributed linear system and thus they should be estimated. Alternatively, sometimes the system description may change slightly from the training to the testing phase. Therefore, it is essential to study the impact of the inaccurate knowledge of these matrices on the trajectory.

Consider a network system on a graph $\mathcal{G}$ with the linear dynamics (5) and described by the set of matrices $\mathcal{D}$. Assume that these matrices are unknown and, instead, access to estimates of these matrices is provided. These estimates are denoted by $\hat{\mathcal{D}} = \{\hat{\mathbf{S}}, \hat{\mathbf{A}}, \hat{\bar{\mathbf{A}}}, \hat{\mathbf{B}}, \hat{\bar{\mathbf{B}}}\}$ where $\hat{\mathbf{S}} \in \mathbb{R}^{N \times N}$ is the estimate of the support matrix (i.e. the exact graph support is unknown), $\hat{\mathbf{A}} \in \mathbb{R}^{N \times N}$ and $\hat{\bar{\mathbf{A}}} \in \mathbb{R}^{F \times F}$ are the estimates of the system matrices, and $\hat{\mathbf{B}} \in \mathbb{R}^{N \times N}$ and $\hat{\bar{\mathbf{B}}} \in \mathbb{R}^{G \times F}$ are the estimates of the control matrices. It is evident that the trajectory $\{\hat{\mathbf{X}}(t)\}$ on the linear dynamical network $\hat{\mathcal{D}}$ could be noticeably different from $\{\mathbf{X}(t)\}$, the one obtained from the system described by $\mathcal{D}$.

The goal is to characterize how the difference in the systems $\mathcal{D}$ and $\hat{\mathcal{D}}$ impacts their respective trajectories $\{\mathbf{X}(t)\}$ and $\{\hat{\mathbf{X}}(t)\}$. Towards this end, a notion of distance between the system matrices is first defined.

Definition 2 (Distance between systems)**.** Given the system matrices $\mathcal{D}$ and $\hat{\mathcal{D}}$, the *distance between system descriptions* is defined as

$$\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}) = \varepsilon, \tag{23}$$

where $\varepsilon > 0$ is the smallest number such that

$$\begin{aligned} &\|\mathbf{S} - \hat{\mathbf{S}}\|_2 \leq \varepsilon \quad \|\mathbf{A} - \hat{\mathbf{A}}\|_2 \leq \varepsilon \quad \|\bar{\mathbf{A}} - \hat{\bar{\mathbf{A}}}\|_\infty \leq \varepsilon, \\ &\qquad \|\mathbf{B} - \hat{\mathbf{B}}\|_2 \leq \varepsilon \quad \|\bar{\mathbf{B}} - \hat{\bar{\mathbf{B}}}\|_\infty \leq \varepsilon. \end{aligned} \tag{24}$$

In other words, Definition 2 determines the distance between two system descriptions as the maximum norm difference in the constitutive matrix norms, with matrices on the graph domain being determined by the spectral norm $\|\cdot\|_2$, and matrices on the feature domain being determined by the infinity norm $\|\cdot\|_\infty$.

First, a result on how the input-state stability of the closed-loop system is affected by the distance between $\mathcal{D}$ and $\hat{\mathcal{D}}$ is obtained.

Proposition 2 (Change in input-state stability)**.** *Consider two systems described by the sets of matrices $\mathcal{D}$ and $\hat{\mathcal{D}}$. Let these systems be controlled by a GNN (17) consisting of L layers of filters $\mathrm{H}_\ell(\cdot; \cdot, \mathcal{H})$ with F_ℓ features and K_ℓ filter taps each. Let the nonlinearity $\sigma(\cdot)$ be such that $|\sigma(a) - \sigma(b)| \leq |a - b|$ and $\sigma(0) = 0$. Then, it holds that*

$$\left|\xi - \hat{\xi}\right| \leq \hat{C}_\xi \ \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}), \tag{25}$$

where $\xi = \xi(\mathcal{D}, \mathcal{H})$ and $\hat{\xi} = \xi(\hat{\mathcal{D}}, \mathcal{H})$ are the stability constants of the system $\mathcal{D}$ and $\hat{\mathcal{D}}$, respectively, and where

$$\hat{C}_\xi = \|\mathbf{A}\|_2 + \|\hat{\bar{\mathbf{A}}}\|_\infty + C_\Phi\big(\|\mathbf{B}\|_2 + \|\hat{\bar{\mathbf{B}}}\|_\infty\big), \tag{26}$$

with $C_\Phi = \prod_{\ell=1}^{L} C_{\mathrm{H}_\ell}$ for C_{H_ℓ} the size of the ℓ^{th} filter, see (13).

Proof. See Appendix B. □

Proposition 2 states that the difference in the stability constants between the system $\mathcal{D}$ and its estimate $\hat{\mathcal{D}}$ depends on the distance $\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$ between them, on the system matrices of both $\mathcal{D}$ and $\hat{\mathcal{D}}$, and on the learned filters through C_Φ. If the matrix description of $\mathcal{D}$ is inaccessible, then Def. 2 can be leveraged to replace $\|\mathbf{A}\|_2$

and $\|\mathbf{B}\|_2$ in (26) by the upper bounds $\|\mathbf{A}\|_2 \leq \|\hat{\mathbf{A}}\|_2 + \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$ and $\|\mathbf{B}\|_2 \leq \|\hat{\mathbf{B}}\|_2 + \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$, respectively. The same holds if $\hat{\mathcal{D}}$ is not known but $\mathcal{D}$ is. It is also noted that, for the case when $F = G = 1$, it follows from the proof that $\hat{C}_\xi = 1 + C_\Phi$ and the bound is proportional to the distance $\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$; see Appendix B.

Next, the goal is to characterize the deviation in the trajectories, namely $\|\mathbf{X}(t) - \hat{\mathbf{X}}(t)\|$, as a function of how different the systems $\mathcal{D}$ and $\hat{\mathcal{D}}$ are. In this context, a controller Φ is acceptable if the resulting closed-loop trajectories of two different systems are similar as long as the systems themselves are similar. This is the case for GNN-based distributed controllers as shown next.

Theorem 3 (Bound on trajectory deviation)**.** *Consider two systems described by the sets of matrices $\mathcal{D}$ and $\hat{\mathcal{D}}$. Let these systems be controlled by a GNN (17) consisting of L layers of filters $\mathrm{H}_\ell(\cdot;\cdot,\mathcal{H})$ with F_ℓ features and K_ℓ filter taps each. Let the nonlinearity $\sigma(\cdot)$ be such that $|\sigma(a) - \sigma(b)| \leq |a - b|$ and $\sigma(0) = 0$. Then, it holds that*

$$\left\| \mathbf{X}(t) - \hat{\mathbf{X}}(t) \right\| \leq \hat{C}_\Phi \hat{C}_t \|\mathbf{X}(0)\| \ \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}), \tag{27}$$

with $\hat{C}_\Phi = \hat{C}_\xi + C_\Phi \Gamma_\Phi \|\hat{\mathbf{B}}\|_2 \|\hat{\mathbf{B}}\|_\infty (1 + 8\sqrt{N})$ for $\hat{C}_\xi$ as in (26), $C_\Phi = \prod_{\ell=1}^{L} C_{\mathrm{H}_\ell}$ and $\Gamma_\Phi = \sum_{\ell=1}^{L} (\Gamma_{\mathrm{H}_\ell}/C_{\mathrm{H}_\ell})$ for C_{H_ℓ} and Γ_{H_ℓ} the size and Lipschitz constant of the ℓ^{th} filter, respectively, see (13) and (15); and with $\hat{C}_t$ such that $\hat{C}_0 = 0$ and

$$\hat{C}_t = t \max\{\xi, \hat{\xi}\}^{t-1} \tag{28}$$

for $t \geq 1$, where ξ and $\hat{\xi}$ are the stability constants of the systems $\mathcal{D}$ and $\hat{\mathcal{D}}$, respectively, as in (22).

Proof. See Appendix C. □

Theorem 3 states that, for a linear dynamical network system under a GNN-based distributed controller, the change in trajectory between the system $\mathcal{D}$ and its estimated description $\hat{\mathcal{D}}$ depends on the value of $\hat{C}_\Phi$ that is independent of time, on the value of $\hat{C}_t$ that is time-varying, and on their distance $\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$. The value of $\hat{C}_\Phi$ is affected by the given system (through matrices in the estimated system $\hat{\mathcal{D}}$ and the number of nodes N) and the resulting trained filters in the GNN (through C_Φ and Γ_Φ). The value of $\hat{C}_t$ is determined by the stability constants ξ and $\hat{\xi}$, and becomes larger as time passes if $\max\{\xi, \hat{\xi}\} \geq 1$, but otherwise decreases for large t. Recall that ξ can be estimated from $\hat{\xi}$ by leveraging Proposition 2. It is noted that the constants $\hat{C}_\Phi$ and $\hat{C}_t$ can be affected by judicious training. For example, by penalizing the size of the filters C_{H_ℓ} and their Lipschitz constant Γ_{H_ℓ} during training, the learned GNN-based controller can be forced to be more stable, see Section 5 for concrete examples.

For the particular case when the closed-loop system and its estimate are guaranteed to be input-state stable, the following corollary can be stated.

Corollary 4 (Bound on trajectory deviation for stable systems)**.** *Consider a system $\mathcal{D}$ and its estimate $\hat{\mathcal{D}}$ such that both satisfy Theorem 1. Then, it holds that*

$$\left\| \mathbf{X}(t) - \hat{\mathbf{X}}(t) \right\| \leq \hat{C} \|\mathbf{X}(0)\| \ \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}), \tag{29}$$

where $\hat{C} = -e^{-1} \hat{C}_\Phi / (\max\{\xi, \hat{\xi}\} \times \log(\max\{\xi, \hat{\xi}\}))$ and $\hat{C}_\Phi$ is given in Theorem 3. Furthermore, it holds that

$$\lim_{t \to \infty} \left\| \mathbf{X}(t) - \hat{\mathbf{X}}(t) \right\| = 0. \tag{30}$$

Proof. See Appendix C. □

It follows from Corollary 4 that if a system and its estimate are guaranteed to be input-state stable, then the trajectory deviation between both systems is bounded by a constant that is proportional to the distance between them and is independent of time t. Furthermore, this deviation is guaranteed to go to zero as t increases.

5. Numerical experiments

In this section, numerical simulations illustrate the performance of GNN-based controllers in a distributed linear-quadratic problem. In particular, problem (7) is solved with $F = G = 1$ so that $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ become scalars that are subsumed into matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

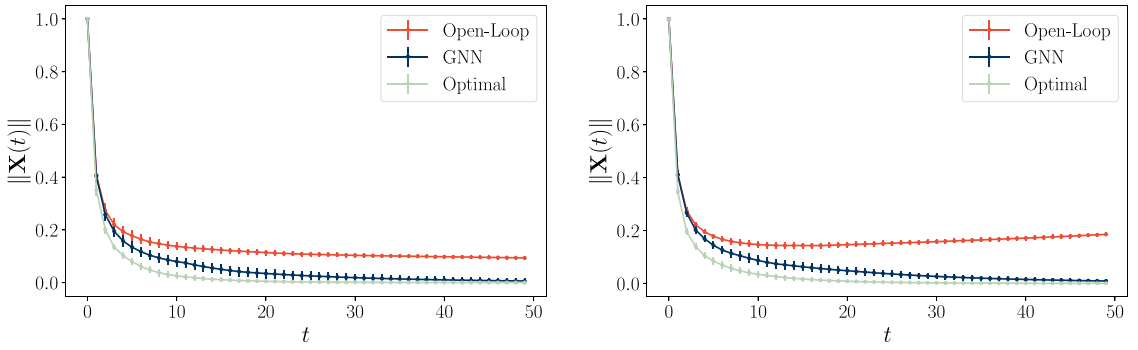
Problem setup. The system has N nodes placed uniformly at random on the $[0, 1] \times [0, 1]$ plane. Edges are drawn between the 5-nearest neighbors of each node. The support matrix $\mathbf{S}$ is considered to be the adjacency matrix, normalized by the largest eigenvalue so that $\|\mathbf{S}\|_2 = 1$. The network system matrix $\mathbf{A}$ and network control matrix $\mathbf{B}$ share the same eigenvectors with $\mathbf{S}$ and the diagonal elements are chosen randomly with a standard Gaussian distribution and are normalized so that $\|\mathbf{A}\|_2 = 0.995$ and $\|\mathbf{B}\|_2 = 1$. The cost matrices are set to $\mathbf{Q} = \mathbf{R} = \mathbf{I}$. Trajectories of length $T = 50$ are simulated. Unless otherwise specified, the networks have $N = 50$ nodes.

Controllers. Five controllers are studied. (i: Optim) The optimal centralized controller is used as a baseline [3, eq. (2.4–8)]. (ii: MLP) A centralized controller can be learned by using a multi-layer perceptron (MLP) with NF_{MLP} units in the hidden layer, and N units in the readout layer [15]. (iii: D-MLP) As a comparative method, the learnable, distributed controller proposed in [16] is used; recall that this method learns a separate MLP for each node, particularly a hidden layer with $F_{\text{D-MLP}}$ units and a single output unit to estimate the control action of the node. (iv: GNN) A two-layer GNN (17) with F_1 features and K_1-order polynomials for the first layer and $F_2 = 1$ and $K_2 = 0$ for the second layer. (v: GF) A K_1-order polynomial graph filter with F_1 features (11), followed by a readout layer which is another graph filter with $F_2 = 1$ output features and $K_2 = 0$ filter taps, see [11]. For the nonlinear methods (ii)-(iv), the function tanh is applied pointwise between the first and the second layers.

Training and evaluation. The controllers (ii)-(v) are trained by solving the equivalent ERM problem (19) over a generated training set consisting of $|\mathcal{T}| = 500$ initial states. The ADAM algorithm [30] with the learning rate μ and forgetting factors 0.9 and 0.999 is used to update the gradients over batches of 20 trajectories. A validation stage leveraging a set of 50 new, independent initial states is computed every 5 training updates. After 30 epochs of training, the parameters that exhibited the best performance during the validation stage are retained. The controllers are evaluated by computing the quadratic cost over trajectories obtained from a set of 50 new, independent initial states. For ease of exposition, the resulting cost is normalized by the lower bound for the distributed linear-quadratic problem obtained in [9]. The training and evaluation process is repeated for 100 different realizations of the system matrices $\mathcal{D}$. Median and standard deviation values of the normalized cost are reported.

Experiment 1: Design hyperparameters. The first experiment studies the performance of the controllers (iv: GNN) and (v: GF) as a function of the number of features at the output of the first layer $F \in \{16, 32, 64\}$, and the order of the polynomial $K \in \{2, 3, 4\}$. The learning rate is chosen from the set $\mu \in \{0.005, 0.01, 0.05\}$ and the one yielding the best performance for each architecture is shown in Table 1. In general, the performance does not vary significantly as a function of the hyperparameters, with a difference of 3.8 percentage points for (iv: GNN) and 5.4 for (v: GF). From now on, the hyperparameter values are set to $F_1 = 16$, $K_1 = 4$ and $\mu = 0.01$ for (iv: GNN), and $F_1 = 64$, $K_1 = 4$ and $\mu = 0.005$ for (v: GF). The fact that $K_1 = 4$ exhibits the best performance for both controllers evidences the importance of repeated communication with one-hop neighbors for collecting information farther away.

Experiment 2: Comparison. For the second experiment, the performance of the controllers (iv: GNN) and (v: GF) is compared



(a) Stable open-loop system (b) Unstable open-loop system

Fig. 1. Comparison with the open-loop system, showing the norm of the evolution of the state norm $\|\mathbf{X}(t)\|$ as a function of time t. (a) This is the case when the system is open-loop stable, i.e. $\|\mathbf{A}\|_2 = 0.995$. It is observed that, while the trajectory is going to zero even in the absence of a controller (open-loop), the use of a GNN-based controller drives the state faster to zero. (b) Consider now an unstable open-loop system given by $\|\mathbf{A}\|_2 = 1.01$. It is observed that the state does not go to zero in the absence of a controller, and that the GNN-based controller successfully drives the state to 0.

Table 1
Normalized cost of the distributed controllers. (a) Distributed controller (iv: GNN) for $\mu = 0.01$. (b) Distributed controller (v: GF) for $\mu = 0.005$. Lower bound: 65(±2).

F/K	2	3	4
16	1.1396(±0.0379)	1.1311(±0.0338)	**1.1052(±0.0295)**
32	1.1440(±0.0348)	1.1286(±0.0275)	1.1354(±0.0255)
64	1.1409(±0.0356)	1.1300(±0.0272)	1.1196(±0.0323)

(a) GNN (iv: GNN)

F/K	2	3	4
16	1.1716(±0.0319)	1.1449(±0.0331)	1.1295(±0.0289)
32	1.1609(±0.0291)	1.1385(±0.0358)	1.1233(±0.0285)
64	1.1466(±0.0361)	1.1248(±0.0313)	**1.1175(±0.0251)**

(b) Graph Filter (v: GF)

to that of the centralized baselines (i: Optim) and (ii: MLP), and that of the distributed method (iii: D-MLP). The hyperparameters of (ii: MLP) and (iii: D-MLP) are set to $(F_{\text{MLP}}, \mu) = (16, 0.005)$ and $(F_{\text{D-MLP}}, \mu) = (16, 0.01)$, respectively, chosen for yielding the best performance from the set $\{16, 32, 64\}$ for the features and $\{0.005, 0.01, 0.05\}$ for the learning rate. The controller (ii: MLP) learns 80,000 parameters and the controller (iii: D-MLP) learns 3,200, while (iv: GNN) learns 80 parameters and (v: GF) learns 320. The centralized controllers (i: Optim) and (ii: MLP) exhibit a normalized cost of 0.9961(±0.0001) and 0.9969(±0.0003), respectively. This shows that these two controllers are better than any possible distributed one. The distributed method (iii: D-MLP) yields a cost of 1.0999(±0.0167), 0.5 percentage points better than (iv: GNN) which shows a cost of 1.1052(±0.0295) and 1.7 percentage points better than (v: GF) which shows a cost of 1.1175(±0.0251). Overall, as expected, the centralized controllers perform better than the distributed ones. The performance of the controller (iii: D-MLP) is slightly better than (iv: D-MLP), possibly due to the fact that (iii: D-MLP) exhibits a larger representation space that can be successfully navigated given the rich training setting available in this simulation. It is observed in experiments 3 and 4, however, that this controller is not robust to changes in the underlying topology nor scales well, precisely due to the large number of parameters. Finally, it is observed that the nonlinear distributed controllers (iii) and (iv) outperform the linear one (v: GF).

Experiment 3: Comparison with open-loop systems. In the third experiment, a comparison with an open-loop system is carried out. It is noted that, from choosing $\|\mathbf{A}\|_2 = 0.995$, the resulting system is open-loop stable and, thus, the state will be driven to zero even in the absence of a controller. In this context, the effect of the distributed controller should be such that it drives the states to zero faster than the open-loop case. The results shown in Fig. 1a indicate that the use of a GNN controller drives the state to zero faster than the open-loop, uncontrolled, system. This illustrates that the GNN controller is better than using no controller, also in the case where the open-loop system is already stable. This is also shown in the resulting cost, which for the open-loop system is 1.5961(±0.0837) while for the GNN controller is 1.1104(±0.0334).

Alternatively, the case of a system that is open-loop unstable is also considered. In this case, the norm of the system matrix is $\|\mathbf{A}\|_2 = 1.01$. It is immediately observed in Fig. 1b that while the open-loop system tends to be unstable (the norm of $\|\mathbf{X}(t)\|$ grows as t grows), the GNN controller effectively drives the state to zero.

More generally, an experiment of the normalized cost as a function of $\|\mathbf{A}\|_2$ is run. This experiment helps visualize the transition between systems that are open-loop stable and systems that are not. The norm of the system matrix $\|\mathbf{A}\|_2$ varies from 0.95 to 1.01. Results are shown in Fig. 2. It is evident that as $\|\mathbf{A}\|_2$ grows, the cost increases, showing that the system is increasingly harder to control. But, while the open-loop system cost seems to exponentially grow, the GNN controller manages to keep the cost low and, as seen in Fig. 1b it effectively drives the state to zero.

Experiment 4: Unknown system matrices. In the fourth experiment, the impact of an unknown system on both the stability (Prop. 2) and the trajectory deviation (Thm. 3) is studied. The controllers are trained on a system $\mathcal{D}$, and then tested on another system $\hat{\mathcal{D}}$ that is a random Gaussian noise perturbation such that $\mathsf{d}(\mathcal{D}, \hat{\mathcal{D}}) = \varepsilon$ for some predefined value of ε. It is observed in (25) that the change in stability is controlled by $C_\Phi = C_{\mathsf{H}_1} C_{\mathsf{H}_2}$, while (27) shows that the trajectory deviation can be controlled by lowering the value of the Lipschitz constants $\{\Gamma_{\mathsf{H}_1}, \Gamma_{\mathsf{H}_2}\}$ and of the size $\{C_{\mathsf{H}_1}, C_{\mathsf{H}_2}\}$ of the filters involved. Therefore, the controller (iv: GNN) is trained with three different penalties: a penalty on the size C_Φ, i.e. the objective function is $\mathsf{J}(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}) + C_\Phi$, a penalty on the Lipschitz constants, i.e. $\mathsf{J}(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}) + (\Gamma_{\mathsf{H}_1} + \Gamma_{\mathsf{H}_2})$, or a penalty on both the filter size and the Lipschitz constant, i.e. $\mathsf{J}(\{\mathbf{X}(t)\}, \{\mathbf{U}(t)\}) + 0.5(\Gamma_{\mathsf{H}_1} + \Gamma_{\mathsf{H}_2} + C_\Phi)$. This is indicated by the legend 'GNN w/ size', 'GNN w/ Lipschitz', and 'GNN w/ both', respectively. The GCNN is also trained without penalties, for comparison, and labeled 'GNN'.

The results are shown on Fig. 3. First, the effects of the unknown system on the stability are analyzed, see Prop. 2. Fig. 3a shows that when training the GNN with a size penalty, the controller leads to a stable closed-loop system 100% of the time for $\varepsilon < 0.05$, fails to control only 0.5% of the trajectories for $\varepsilon = 0.0562$ and 10% of the trajectories for $\varepsilon = 0.1$. When training with both

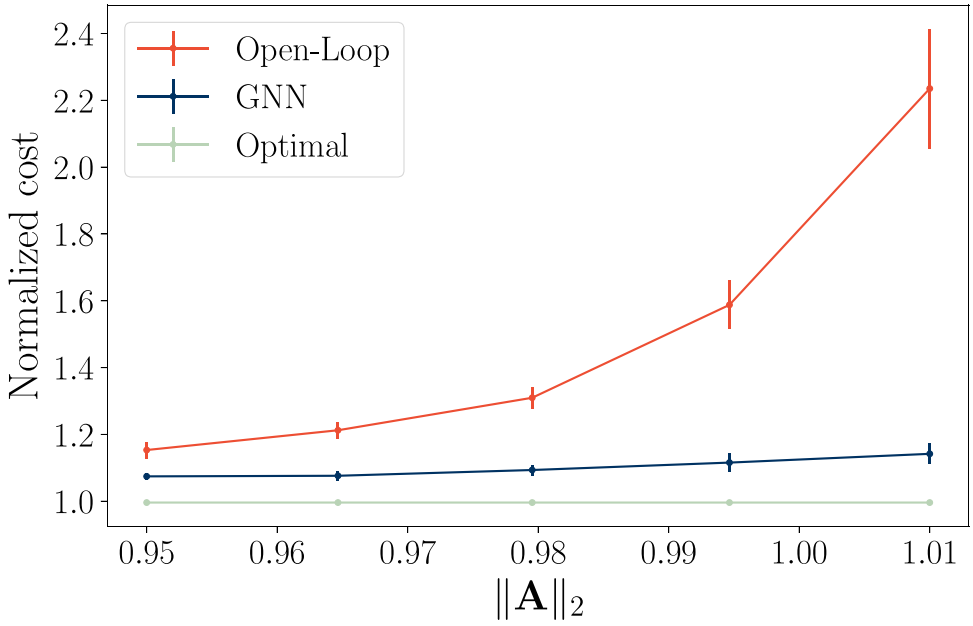
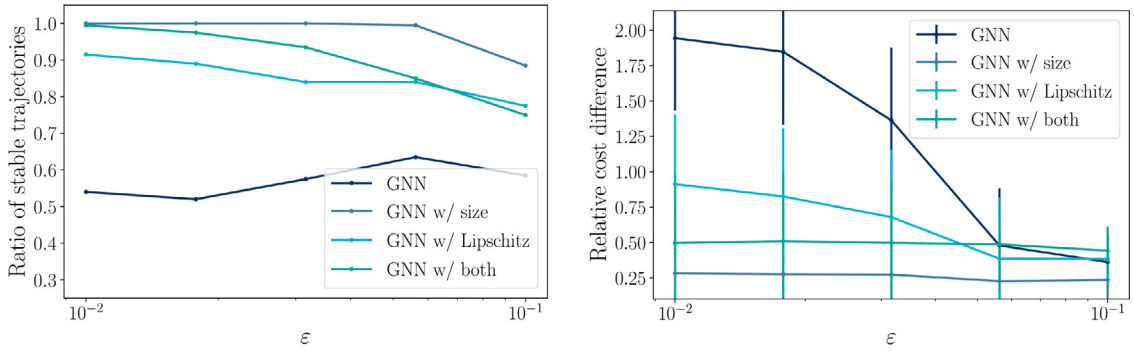


Fig. 2. Normalized cost as a function of the norm of $\|\mathbf{A}\|_2$. It is observed that the cost for the uncontrolled, open-loop system, grows exponentially with the norm of $\|\mathbf{A}\|_2$ as expected. The cost of the GNN-controller, however, grows only slightly with increasing values of $\|\mathbf{A}\|_2$.



(a) Stability (b) Trajectory deviation

Fig. 3. Simulation results for a network with unknown system matrices as a function of the distance ε between the systems, see (23). (a) Ratio of stable trajectories as a function of ε; it is observed that when training with a penalty on the size C_Φ of the GNN, the resulting trajectories are stable for larger values of ε. (b) Cost difference of the controlled trajectories relative to the cost on the perfectly known system; it is observed that when training with a penalty on the size C_Φ of the GNN, the resulting controller achieves the lowest relative cost difference. The distributed controller (iii: D-MLP) and the centralized controller (ii: MLP) are not shown since they exhibit relative cost differences of approximately 7.5 and 1400, respectively, thus being out of scale; this is likely to their failure to control trajectories.

penalties, the controller is able to lead to stable systems 100% of the time for $\varepsilon = 0.01$, but then decays rapidly in its ability to stabilize the system as ε grows. Training with Lipschitz penalty only leads to a controller that can stabilize about 92% of the trajectories for $\varepsilon = 0.01$ and then falls to stabilizing about 80% of the trajectories for $\varepsilon = 0.1$. This shows that training with a penalty on the size C_Φ of the GNN has the most impact on the ability of the learned distributed controller to stabilize the system, as predicted by Prop. 2 Finally, note that when training the GNN without penalties, the resulting controller stabilizes only 55% of the trajectories on an unknown system.

It is observed in Fig. 3b the relative difference between the cost obtained when testing on the system $\mathcal{D}$ and that obtained when testing on system $\hat{\mathcal{D}}$ for different values of system distance ε among stable trajectories. First, it is noted that training with a penalty on the size of the GNN leads to a controller that is unaffected by changes in the system, exhibiting a relative cost difference of 0.25 for all values of ε under study. The other three controllers seem to improve in their relative difference as ε grows, and this can be explained because the cost is being computed only among stable trajectories. This implies that, while ε grows and less trajectories are being stabilized, the ones that remain do achieve good relative cost difference. Finally, it is noted that the distributed controller (iii: D-MLP) and the centralized learnable controller (ii: MLP) were also considered in this simulation. These controllers exhibited relative differences of approximately 7.5 and 1400, respectively, thus falling out of scale and not being shown in the figures. This results show that neither the (iii: D-MLP) nor the (ii: MLP) controllers are robust to changes in the system dynamics.

Experiment 5: Scalability. In the last experiment, scalability of the distributed controllers (iii)-(v) is compared. These methods are trained on a system with $N = 50$ nodes, and then at test time, are used on increasingly larger systems $N \in \{50, 63, 75, 87, 100\}$. The resulting costs of the stable trajectories are shown in Fig. 4. It is observed that, while the D-MLP performs better when tested on the same system as it was trained (see experiment 2), it does not transfer as well to larger systems. This is likely to be because it assigns a different fully connected neural network controller to

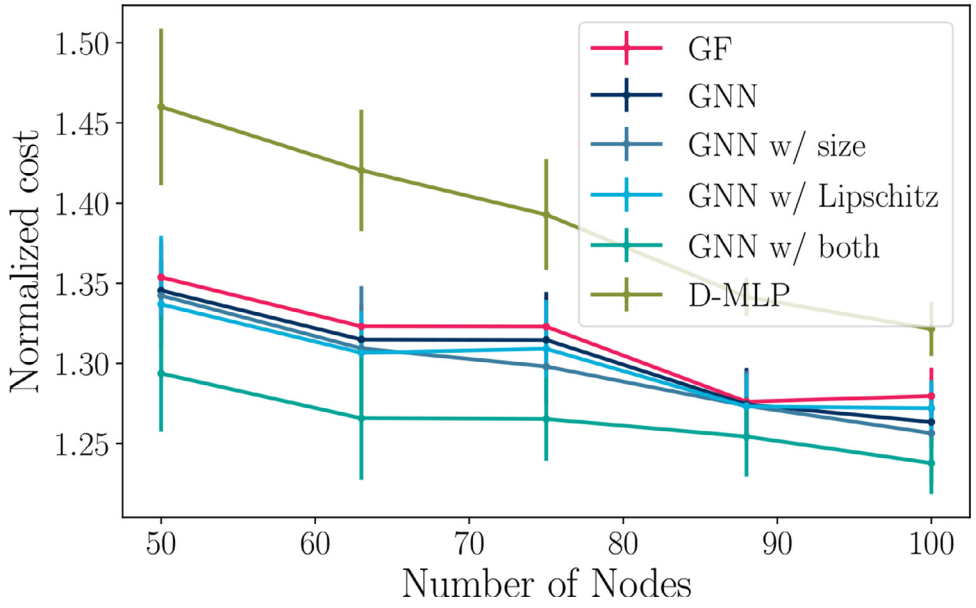


Fig. 4. Normalized cost for the stable trajectories of a GNN-based controller trained on 50 nodes and tested on a larger network system. It is observed that training with penalties on both the Lipschitz constant and the size of the filters lead to best scalability results.

each component, so that, when tested on larger systems, it has to replicate this controller on other nodes and that may have a substantially different topological neighborhood. Controllers (iv: GNN) and (v: GF), on the other hand, successfully adapt to larger systems, even when trained on small ones. In particular, training with penalties on both the Lipschitz constant and the size of the filters leads to the best scalability results. It is noted that the centralized controller (ii: MLP) cannot transfer to systems with different number of nodes since the number of learned parameters depends on the number of nodes.

6. Conclusion

This paper proposes to address the issue of the intractability of distributed optimal controllers by leveraging a nonlinear GNN-based parametrization. While the resulting controller is suboptimal, it exhibits several desirable properties such as distributed computation, efficiency and scalability. These controllers are applied to the distributed linear-quadratic problem, which can be cast as a self-supervised empirical risk minimization problem, and then solved by means of machine learning techniques. A sufficient condition for the resulting closed-loop system to be input-state stable is derived in terms of the filter taps of the GNN-based controller. Additionally, the trajectory deviation due to mismatch of the system descriptions is shown to also be controlled by the filter taps. Extensive simulations illustrate the satisfactory performance exhibited by GNN-based controllers as well as the ability to be trained to exhibit certain desirable characteristics such as an improved closed-loop stability or a smaller trajectory deviation under model mismatch. The resulting controller is also shown to scale to larger systems. Future research on the topic may involve the study of equilibrium points of a GNN-controlled system and their Lyapunov stability, the use of distributed optimization techniques to solve the self-supervised learning problem, and the adoption of other non-convolutional GNN-based architectures.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Institutions: University of California, Berkeley

Co-authors in previous works: Santiago Segarra, Geert Leus, Elvin Isufi, Joan Bruna, Giorgos Giannakis, Weiyu Huang, Vassilis Ioannidis, Aryan Mokhtari, Luiz Chamon, Alec Koppel, Javad Lavaei, Nikolai Matni.

Appendix A. Auxiliary Results

In this appendix four Lemmas that are useful for proving the theorems and propositions of Sections Appendix B and Appendix C are included. The first two Lemmas establish an upper bound on the output of a graph filter (Lemma 5) and a GNN (Lemma 6) as a function of the size of the filters involved. The following two lemmas determine the Lipschitz continuity with respect to the support matrix $\mathbf{S}$ of the graph filter (Lemma 7) and the GNN (Lemma 8) as a function of the filter sizes and the Lipschitz constants.

Lemma 5 (Bound on Graph Filter Output)**.** *Let* $\mathsf{H}: \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$ *be a graph filter (11) defined over a support matrix* $\mathbf{S} \in \mathbb{R}^{N\times N}$*. Let* $\mathbf{X} \in \mathbb{R}^{N\times F}$ *be any graph signal such that* $\|\mathbf{X}\| < \infty$*. Then,*

$$\left\|\mathsf{H}(\mathbf{X};\mathbf{S},\mathcal{H})\right\| \leq C_{\mathsf{H}} \left\|\mathbf{X}\right\|, \tag{A.1}$$

with C_{H} *being the size of the filter bank, see (13).*

Proof. Recall that the norm associated to the graph signal space is given by the $L_{2,1}$ entrywise matrix norm, see (9). Then, the graph signal size of the output $\mathbf{Y} = \mathsf{H}(\mathbf{X};\mathbf{S},\mathcal{H})$ can be computed as

$$\|\mathbf{Y}\| = \sum_{g=1}^{G} \|\mathbf{y}^g\|_2 = \sum_{g=1}^{G} \left\| \sum_{f=1}^{F} \mathbf{H}_{fg}(\mathbf{S})\mathbf{x}^f \right\|_2, \tag{A.2}$$

where $\mathbf{H}_{fg}(\mathbf{S}) = \sum_{k=0}^{K} [\mathbf{H}_k]_{fg} \mathbf{S}^k$, see (13), and where $\|\mathbf{x}\|_2$ represents the Euclidean norm on vectors. One can apply the triangular inequality to (A.2) to obtain:

$$\|\mathbf{Y}\| \leq \sum_{g=1}^{G} \sum_{f=1}^{F} \left\| \mathbf{H}_{fg}(\mathbf{S})\mathbf{x}^f \right\|_2 \tag{A.3}$$

and noticing that the summation is comprised of Euclidean vector norms, the submultiplicativity of the corresponding matrix spectral

norm can be used to arrive at

$$\|\mathbf{Y}\| \leq \sum_{g=1}^{G} \sum_{f=1}^{F} \left\|\mathbf{H}_{fg}(\mathbf{S})\right\|_2 \left\|\mathbf{x}^f\right\|_2, \tag{A.4}$$

which, noting that the sum over g only affects $\|\mathbf{H}_{fg}(\mathbf{S})\|_2$, can be rearranged as

$$\|\mathbf{Y}\| \leq \sum_{f=1}^{F} \left\|\mathbf{x}^f\right\|_2 \sum_{g=1}^{G} \left\|\mathbf{H}_{fg}(\mathbf{S})\right\|_2. \tag{A.5}$$

Next, note that $\sum_{g=1}^{G} \|\mathbf{H}_{fg}(\mathbf{S})\|_2$ is the sum of all the spectral norms of the filters along the g dimension, thus the result is a scalar that depends on f and is denoted with C_f in this proof, i.e. $\sum_{g=1}^{G} \|\mathbf{H}_{fg}(\mathbf{S})\|_2 = C_f$. For each value of f, there is a different C_f, and it holds true that $C_f \leq \sup_{f=1,\ldots,F}$. This implies that $\sum_{g=1}^{G} \|\mathbf{H}_{fg}(\mathbf{S})\|_2 \leq \sup_{f=1,\ldots,F} \sum_{g=1}^{G} \|\mathbf{H}_{fg}(\mathbf{S})\|_2$.

From (13), note that each element of the matrix $\mathbf{C}_{\mathsf{H}} \in \mathbb{R}^{F\times G}$ is given by $\max_{\lambda\in[\lambda_l,\lambda_h]} |h_{fg}(\lambda)|$ for some chosen values of $[\lambda_l, \lambda_h]$. Then, if λ_l and λ_h are the minimum and maximum eigenvalues of $\mathbf{S}$ as is usually the case, then it follows that $\sup_{f=1,\ldots,F} \sum_{g=1}^{G} \|\mathbf{H}_{fg}(\mathbf{S})\|_2 \leq \|\mathbf{C}_{\mathsf{H}}\|_\infty = C_{\mathsf{H}}$, see (13). Recall that $\|\mathbf{A}\|_\infty$ is the infinity norm of matrices (i.e. maximum absolute row sum). Finally, (A.5) can be upper bounded as

$$\|\mathbf{Y}\| \leq C_{\mathsf{H}} \sum_{f=1}^{F} \|\mathbf{x}^f\|_2. \tag{A.6}$$

Noting that $\sum_{f=1}^{F} \|\mathbf{x}^f\|_2 = \|\mathbf{X}\|$ completes the proof. □

Lemma 6 (Bound on GNN Output). *Let $\Phi(\cdot; \mathbf{S}, \mathcal{H}) : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$ be a GNN (17) with L layers defined over a support matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$. Let the nonlinearity $\sigma(\cdot)$ be such that $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, for some $C_\sigma > 0$. Then, for every graph signal $\mathbf{X} \in \mathbb{R}^{N\times F}$ with $\|\mathbf{X}\| < \infty$, it holds that*

$$\left\|\Phi(\mathbf{X}; \mathbf{S}, \mathcal{H})\right\| \leq C_\sigma^L C_\Phi \left\|\mathbf{X}\right\|, \tag{A.7}$$

where $C_\Phi = \prod_{\ell=1}^{L} C_{\mathsf{H}_\ell}$ for C_{H_ℓ} the size of the ℓ^{th} filter, see (13).

Proof. Consider the computation of layer ℓ

$$\mathbf{X}_\ell = \sigma\Big(\mathsf{H}_\ell\big(\mathbf{X}_{\ell-1}; \mathbf{S}, \mathcal{H}_\ell\big)\Big), \tag{A.8}$$

whose norm is given by (9),

$$\|\mathbf{X}_\ell\| = \sum_{g=1}^{F_\ell} \|\mathbf{x}_\ell^g\|_2, \tag{A.9}$$

with

$$\mathbf{x}_\ell^g = \sigma\Big(\sum_{f=1}^{F_{\ell-1}} \mathbf{H}_{\ell fg}(\mathbf{S}) \mathbf{x}_{\ell-1}^f\Big), \tag{A.10}$$

where $\mathbf{H}_{\ell fg}(\mathbf{S}) = \sum_{k=0}^{K_\ell} [\mathbf{H}_{\ell k}]_{fg} \mathbf{S}^k$ denotes the scalar-valued graph filter.

Substituting (A.10) into (A.9) and using the hypothesis on the nonlinearity that $|\sigma(x)| \leq C_\sigma |x|$ for all x, the following upper bound on the norm of the output signal at layer ℓ is obtained:

$$\|\mathbf{X}_\ell\| \leq C_\sigma \sum_{g=1}^{F_\ell} \Big\| \sum_{f=1}^{F_{\ell-1}} \mathbf{H}_{\ell fg}(\mathbf{S}) \mathbf{x}_{\ell-1}^f \Big\|_2, \tag{A.11}$$

which is simply

$$\|\mathbf{X}_\ell\| \leq C_\sigma \left\|\mathsf{H}_\ell(\mathbf{X}_{\ell-1}; \mathbf{S}, \mathcal{H}_\ell)\right\|. \tag{A.12}$$

Now, using Lemma 5 on (A.12) yields

$$\|\mathbf{X}_\ell\| \leq C_\sigma C_{\mathsf{H}_\ell} \|\mathbf{X}_{\ell-1}\|. \tag{A.13}$$

Repeating (A.13) for all consecutive layers until reaching $\ell = 1$ leads to

$$\|\mathbf{X}_\ell\| \leq C_\sigma^\ell \prod_{\ell'=1}^{\ell} C_{\mathsf{H}_{\ell'}} \|\mathbf{X}_0\|. \tag{A.14}$$

By substituting $\ell = L$ into (A.14) and recalling that $\mathbf{X}_0 = \mathbf{X}$, $\Phi(\mathbf{X}; \mathbf{S}, \mathcal{H}) = \mathbf{X}_L$ and $C_\Phi = \prod_{\ell=1}^{L} C_{\mathsf{H}_\ell}$, the proof is completed. □

In what follows, we state two Lemmas regarding the Lipschitz continuity of graph filters and GNNs with respect to the support matrix $\mathbf{S}$. These results have already been correspondingly proved, and are just rewritten here to unify notation.

Lemma 7 (Lipschitz continuity of graph filter with respect to. S). *Let $\mathsf{H} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$ be a graph filter (11). Let $\mathbf{S} \in \mathbb{R}^{N\times N}$ and $\hat{\mathbf{S}} \in \mathbb{R}^{N\times N}$ be two support matrices, such that $\|\mathbf{S} - \hat{\mathbf{S}}\|_2 \leq \varepsilon$. Then, for any graph signal $\mathbf{X} \in \mathbb{R}^{N\times F}$ such that $\|\mathbf{X}\| < \infty$, it holds that*

$$\left\|\mathsf{H}(\mathbf{X}; \hat{\mathbf{S}}, \mathcal{H}) - \mathsf{H}(\mathbf{X}; \mathbf{S}, \mathcal{H})\right\| \leq \varepsilon(1 + 8\sqrt{N})\Gamma_{\mathsf{H}} \|\mathbf{X}\| + \mathrm{O}(\varepsilon^2), \tag{A.15}$$

with Γ_{H} being the Lipschitz constant filter bank, see (14).

Proof. See [13, Thm. 1]. □

Lemma 8 (Lipschitz continuity of the GNN with respect to. S). *Let $\Phi(\cdot; \cdot, \mathcal{H}) : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times G}$ be a GNN (17) with L layers. Let $\sigma(\cdot)$ be such that $|\sigma(x) - \sigma(y)| \leq \Gamma_\sigma |x - y|$ for all $x, y \in \mathbb{R}$ for some $\Gamma_\sigma > 0$, and $\sigma(0) = 0$. Let $\mathbf{S} \in \mathbb{R}^{N\times N}$ and $\hat{\mathbf{S}} \in \mathbb{R}^{N\times N}$ be two support matrices such that $\|\mathbf{S} - \hat{\mathbf{S}}\|_2 \leq \varepsilon$. Then, for every graph signal $\mathbf{X} \in \mathbb{R}^{N\times F}$ with $\|\mathbf{X}\| < \infty$, it holds that*

$$\left\|\Phi(\mathbf{X}; \hat{\mathbf{S}}, \mathcal{H}) - \Phi(\mathbf{X}; \mathbf{S}, \mathcal{H})\right\| \leq \varepsilon(1 + 8\sqrt{N})\Gamma_\sigma^L C_\Phi \sum_{\ell=1}^{L} \frac{\Gamma_{\mathsf{H}_\ell}}{C_{\mathsf{H}_\ell}} \|\mathbf{X}\| + \mathrm{O}(\varepsilon^2), \tag{A.16}$$

where $C_\Phi = \prod_{\ell=1}^{L} C_{\mathsf{H}_\ell}$ for C_{H_ℓ} the size of ℓ^{th} filter, see (13), and where Γ_{H_ℓ} is the corresponding Lipschitz constant, see (15).

Proof. See [13, Thm. 4]. □

Appendix B. Proof of Closed-Loop Stability

In this appendix, we first prove Theorem 1 that gives a sufficient condition for the GNN-controlled system $\mathcal{D}$ to be stable. We then prove Proposition 2 stating how the stability constant ξ changes from system $\mathcal{D}$ to system $\hat{\mathcal{D}}$.

Proof of Theorem 1.. The system dynamics with a GNN-based, exploratory controller given by $\mathbf{U}(t) = \Phi(\mathbf{X}(t); \mathbf{S}, \mathcal{H}) + \mathbf{E}(t)$ are

$$\mathbf{X}(t) = \mathbf{A}\mathbf{X}(t-1)\bar{\mathbf{A}} + \mathbf{B}\Phi(\mathbf{X}(t-1))\bar{\mathbf{B}} + \mathbf{B}\mathbf{E}(t-1)\bar{\mathbf{B}}. \tag{B.1}$$

The graph signal norm of the trajectory can be bounded by applying the triangular inequality as follows:

$$\begin{aligned}\|\mathbf{X}(t)\| \leq{}& \|\mathbf{A}\|_2 \|\bar{\mathbf{A}}\|_\infty \|\mathbf{X}(t-1)\| \\ &+ \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty \|\Phi(\mathbf{X}(t-1))\| + \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty \|\mathbf{E}(t-1)\|.\end{aligned} \tag{B.2}$$

The term $\|\Phi(\mathbf{X}(t); \mathbf{S}, \mathcal{H})\|$ can be bounded by leveraging Lemma 6 on the bound of the output of a GNN as

$$\|\mathbf{U}(t)\| = \left\|\Phi\big(\mathbf{X}(t); \mathbf{S}, \mathcal{H}\big)\right\| \leq C_\Phi \|\mathbf{X}(t)\|, \tag{B.3}$$

with $C_\sigma = 1$. This result is used in (B.2), to yield

$$x_t \leq \xi x_{t-1} + \beta e_{t-1}, \tag{B.4}$$

where $x_t = \|\mathbf{X}(t)\|$, $\xi = \|\mathbf{A}\|_2 \|\bar{\mathbf{A}}\|_\infty + C_\Phi \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty$ is given in (22), $\beta = \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty$ and $e_t = \|\mathbf{E}(t)\|$. By repeatedly applying (B.4), the following inequality is obtained:

$$x_t \leq \xi^t x_0 + \beta \sum_{\tau=0}^{t-1} \xi^\tau e_{t-\tau-1}. \tag{B.5}$$

Now, considering the summation series that defines the stability as in (20), one obtains:

$$\sum_{t=0}^{\infty} x_t \leq x_0 \sum_{t=0}^{\infty} \xi^t + \beta \sum_{t=0}^{\infty} \sum_{\tau=0}^{t-1} \xi^\tau e_{t-\tau-1}. \tag{B.6}$$

Leveraging the assumptions that $\xi < 1$ and $\sum_{t=0}^{\infty} e_t < \infty$, the above inequality yields

$$\sum_{t=0}^{\infty} x_t \leq \frac{x_0}{1-\xi} + \frac{\beta}{1-\xi} \sum_{t=0}^{\infty} e_t, \tag{B.7}$$

where the fact that, under these assumptions, it holds that $\sum_{t=0}^{\infty} \sum_{\tau=0}^{t-1} \xi^\tau e_{t-\tau-1} \leq (\sum_{t=0}^{\infty} e_t)(\sum_{t=0}^{\infty} \xi^t)$ was used. The proof is complete by replacing the definitions of x_t, e_t and β in (B.7). Thus, the system is input-state stable with constants $\beta_0 = \|\mathbf{X}(0)\|/(1-\xi)$ and $\beta_1 = \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty/(1-\xi)$. □

Next, we prove the change in the stability constant when $\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}) = \varepsilon$.

Proof of Proposition 2.. Start by writing the stability constant $\hat{\xi} = \xi(\hat{\mathcal{D}}, \mathcal{H})$ as given by (22) to obtain

$$\hat{\xi} = \hat{\xi}(\hat{\mathcal{D}}, \mathcal{H}) = \|\hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty + C_\Phi \|\hat{\mathbf{B}}\|_2 \|\underline{\hat{\mathbf{B}}}\|_\infty. \tag{B.8}$$

This equation is equivalent to

$$\begin{aligned} \hat{\xi} \quad &= \|\hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty - \|\mathbf{A}\|_2 \|\hat{\mathbf{A}}\|_\infty \\ &\quad + C_\Phi \Big(\|\hat{\mathbf{B}}\|_2 \|\underline{\hat{\mathbf{B}}}\|_\infty - \|\mathbf{B}\|_2 \|\bar{\mathbf{B}}\|_\infty \Big) + \xi. \end{aligned} \tag{B.9}$$

The first term can be rewritten as

$$\begin{aligned} &\|\hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty - \|\mathbf{A}\|_2 \|\bar{\mathbf{A}}\|_\infty \\ &= \big(\|\hat{\mathbf{A}}\|_2 - \|\mathbf{A}\|_2 \big) \|\underline{\hat{\mathbf{A}}}\|_\infty + \|\mathbf{A}\|_2 \big(\|\underline{\hat{\mathbf{A}}}\|_\infty - \|\bar{\mathbf{A}}\|_\infty \big). \end{aligned} \tag{B.10}$$

From the definition of the distance $\mathrm{d}(\mathcal{D}, \hat{\mathcal{D}}) = \varepsilon$ it is known that $-\varepsilon \leq \|\hat{\mathbf{A}}\|_2 - \|\mathbf{A}\|_2 \leq \varepsilon$, and analogously for $\|\bar{\mathbf{A}}\|_\infty$, so that (B.10) can be bounded by

$$\|\hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty - \|\mathbf{A}\|_2 \|\hat{\mathbf{A}}\|_\infty \leq \varepsilon \big(\|\mathbf{A}\|_2 + \|\underline{\hat{\mathbf{A}}}\|_\infty \big). \tag{B.11}$$

Following the same reasoning for the control matrices, one obtains:

$$\|\hat{\mathbf{B}}\|_2 \|\underline{\hat{\mathbf{B}}}\|_\infty - \|\mathbf{B}\|_2 \|\hat{\mathbf{B}}\|_\infty \leq \varepsilon \big(\|\mathbf{B}\|_2 + \|\underline{\hat{\mathbf{B}}}\|_\infty \big). \tag{B.12}$$

By substituting (B.11) and (B.12) into (B.9) and defining $\hat{C}_\xi = \|\mathbf{A}\|_2 + \|\underline{\hat{\mathbf{A}}}\|_\infty + C_\Phi (\|\mathbf{B}\|_2 + \|\underline{\hat{\mathbf{B}}}\|_\infty)$, the proof is complete. □

Appendix C. Proof of Trajectory Deviations

In this appendix, Theorem 3 bounding the trajectory deviation between systems $\mathcal{D}$ and $\hat{\mathcal{D}}$ is proved. Then, Corollary 4 that considers the special case when both $\mathcal{D}$ and $\hat{\mathcal{D}}$ are input-state stable is also proved.

Proof of Theorem 3.. The dynamic of the error graph signal $\mathbf{X}(t) - \hat{\mathbf{X}}(t)$ is given by

$$\begin{aligned} \mathbf{X}(t) - \hat{\mathbf{X}}(t) = \quad &\mathbf{A}\mathbf{X}(t-1)\bar{\mathbf{A}} - \hat{\mathbf{A}}\hat{\mathbf{X}}(t-1)\underline{\hat{\mathbf{A}}} \\ &+ \mathbf{B}\mathbf{U}(t-1)\bar{\mathbf{B}} - \hat{\mathbf{B}}\hat{\mathbf{U}}(t-1)\underline{\hat{\mathbf{B}}}. \end{aligned} \tag{C.1}$$

The evolution of $\mathbf{X}(t)$ and $\hat{\mathbf{X}}(t)$ and that of $\mathbf{U}(t)$ and $\hat{\mathbf{U}}(t)$ are studied separately.

To study the first part of the right-hand side of (C.1), one can write:

$$\begin{aligned} &\mathbf{A}\mathbf{X}(t-1)\bar{\mathbf{A}} - \hat{\mathbf{A}}\hat{\mathbf{X}}(t-1)\underline{\hat{\mathbf{A}}} = \mathbf{A}\mathbf{X}(t-1)\big(\bar{\mathbf{A}} - \underline{\hat{\mathbf{A}}}\big) \\ &\quad + \big(\mathbf{A} - \hat{\mathbf{A}}\big)\mathbf{X}(t-1)\underline{\hat{\mathbf{A}}} + \hat{\mathbf{A}}\big(\mathbf{X}(t-1) - \hat{\mathbf{X}}(t-1)\big)\underline{\hat{\mathbf{A}}}. \end{aligned} \tag{C.2}$$

Observe that (C.2) consists of three terms containing each of the errors between system matrices and states. Computing the size of the graph signal in (C.2), see (9), and applying the triangular inequality for each of the three terms, one obtains:

$$\begin{aligned} \Big\| \quad &\mathbf{A}\mathbf{X}(t-1)\bar{\mathbf{A}} - \hat{\mathbf{A}}\hat{\mathbf{X}}(t-1)\underline{\hat{\mathbf{A}}} \Big\| \\ &\leq \|\mathbf{A}\|_2 \sum_{f=1}^{F} \big\| \mathbf{x}^f(t-1) \big\|_2 \sum_{g=1}^{F} \big| [\bar{\mathbf{A}}]_{fg} - [\underline{\hat{\mathbf{A}}}]_{fg} \big| \\ &\quad + \big\| \mathbf{A} - \hat{\mathbf{A}} \big\|_2 \sum_{f=1}^{F} \big\| \mathbf{x}^f(t-1) \big\|_2 \sum_{g=1}^{F} \big| [\underline{\hat{\mathbf{A}}}]_{fg} \big| \\ &\quad + \|\hat{\mathbf{A}}\|_2 \sum_{f=1}^{F} \Big\| \mathbf{x}^f(t-1) - \hat{\mathbf{x}}^f(t-1) \Big\|_2 \sum_{g=1}^{F} \big| [\underline{\hat{\mathbf{A}}}]_{fg} \big|. \end{aligned} \tag{C.3}$$

Now, using the bound $\sum_{g=1}^{F} |[\underline{\hat{\mathbf{A}}}]_{fg}| \leq \max_f \sum_{g=1}^{F} |[\underline{\hat{\mathbf{A}}}]_{fg}| = \|\underline{\hat{\mathbf{A}}}\|_\infty$, and analogously for $(\bar{\mathbf{A}} - \underline{\hat{\mathbf{A}}})$, one can write:

$$\begin{aligned} &\Big\| \mathbf{A}\mathbf{X}(t-1)\bar{\mathbf{A}} - \hat{\mathbf{A}}\hat{\mathbf{X}}(t-1)\underline{\hat{\mathbf{A}}} \Big\| \\ &\leq \Big(\|\mathbf{A}\|_2 \|\bar{\mathbf{A}} - \underline{\hat{\mathbf{A}}}\|_\infty + \|\mathbf{A} - \hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty \Big) \|\mathbf{X}(t-1)\| \\ &\quad + \|\hat{\mathbf{A}}\|_2 \|\underline{\hat{\mathbf{A}}}\|_\infty \|\mathbf{X}(t-1) - \hat{\mathbf{X}}(t-1)\|, \end{aligned} \tag{C.4}$$

where the resulting sum over f has been replaced for the corresponding size of the graph signal, see (9).

Proceed analogously to (C.4), the second term in the right-hand side of (C.1) can be bounded as

$$\begin{aligned} &\Big\| \mathbf{B}\mathbf{U}(t-1)\bar{\mathbf{B}} - \hat{\mathbf{B}}\hat{\mathbf{U}}(t-1)\underline{\hat{\mathbf{B}}} \Big\| \\ &\leq \Big(\|\mathbf{B}\|_2 \|\bar{\mathbf{B}} - \underline{\hat{\mathbf{B}}}\|_\infty + \|\mathbf{B} - \hat{\mathbf{B}}\|_2 \|\underline{\hat{\mathbf{B}}}\|_\infty \Big) \|\mathbf{U}(t-1)\| \\ &\quad + \|\hat{\mathbf{B}}\|_2 \|\underline{\hat{\mathbf{B}}}\|_\infty \|\mathbf{U}(t-1) - \hat{\mathbf{U}}(t-1)\|. \end{aligned} \tag{C.5}$$

The control term $\|\mathbf{U}(t)\|$ is a GNN with input $\mathbf{X}(t)$ and can thus be bounded by leveraging Lemma 6, i.e. $\|\mathbf{U}(t)\| \leq C_\Phi \|\mathbf{X}(t)\|$. To bound $\|\mathbf{U}(t) - \hat{\mathbf{U}}(t)\|$, $\Phi(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H})$ is added and subtracted, and the size of the graph signal computed, to obtain

$$\begin{aligned} \Big\| \mathbf{U}(t) - \hat{\mathbf{U}}(t) \Big\| &= \Big\| \Phi\big(\mathbf{X}(t); \mathbf{S}, \mathcal{H}\big) - \Phi\big(\hat{\mathbf{X}}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \Big\| \\ &\leq \Big\| \Phi\big(\mathbf{X}(t); \mathbf{S}, \mathcal{H}\big) - \Phi\big(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \Big\| \\ &\quad + \Big\| \Phi\big(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H}\big) - \Phi\big(\hat{\mathbf{X}}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \Big\|, \end{aligned} \tag{C.6}$$

where the triangular inequality was used. For the first term in (C.6), it follows from Lemma 8 that:

$$\Big\| \Phi(\mathbf{X}(t); \mathbf{S}, \mathcal{H}) - \Phi(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H}) \Big\| \leq \Gamma(\varepsilon) \Gamma_\Phi \|\mathbf{X}(t)\|, \tag{C.7}$$

where $\Gamma(\varepsilon) = (1 + 8\sqrt{N})\varepsilon$ with $\varepsilon = \mathrm{d}(\mathcal{D}, \hat{\mathcal{D}})$ depends on the characteristics of the support matrices $\mathbf{S}$ and $\hat{\mathbf{S}}$, and where $\Gamma_\Phi = C_\Phi \sum_{\ell=1}^{L} \Gamma_{\mathrm{H}_\ell} / C_{\mathrm{H}_\ell}$ depends on the learned filters $\mathrm{H}_\ell(\cdot; \cdot, \mathcal{H})$. To bound the second term in (C.6), recall that the output of a GNN is its value at the last layer

$$\begin{aligned} \Big\| \Phi\big(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \quad &- \Phi\big(\hat{\mathbf{X}}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \Big\| = \Big\| \mathbf{X}_L - \mathbf{X}_L \Big\| \\ &= \Big\| \sigma\big(\mathrm{H}_L(\mathbf{X}_{L-1}; \hat{\mathbf{S}}, \mathcal{H})\big) - \sigma\big(\mathrm{H}_L(\hat{\mathbf{X}}_{L-1}; \hat{\mathbf{S}}, \mathcal{H})\big) \Big\|. \end{aligned} \tag{C.8}$$

Using the assumption that $|\sigma(x) - \sigma(y)| \leq |x - y|$ for all $x, y \in \mathbb{R}$, (C.8) can be upper bounded by

$$\Big\| \Phi\big(\mathbf{X}(t); \hat{\mathbf{S}}, \mathcal{H}\big) - \Phi\big(\hat{\mathbf{X}}(t); \hat{\mathbf{S}}, \mathcal{H}\big) \Big\| \leq \Big\| \mathrm{H}_L(\mathbf{X}_{L-1} - \hat{\mathbf{X}}_{L-1}; \hat{\mathbf{S}}, \mathcal{H}) \Big\|. \tag{C.9}$$

where the linearity of the filter with respect to the input $\mathbf{X}_{L-1}$ was used. Leveraging Lemma 5 on the upper bound of a graph filter, one obtains:

$$\Big\| \mathrm{H}_\ell(\mathbf{X}_{L-1} - \hat{\mathbf{X}}_{L-1}; \hat{\mathbf{S}}, \mathcal{H}) \Big\| \leq C_{\mathrm{H}_L} \Big\| \mathbf{X}_{L-1} - \hat{\mathbf{X}}_{L-1} \Big\|. \tag{C.10}$$

Repeatedly applying (C.9) and (C.10), the following upper bound on

the second term of (C.6) is obtained:

$$\left\|\Phi\big(\mathbf{X}(t);\hat{\mathbf{S}},\mathcal{H}\big)-\Phi\big(\hat{\mathbf{X}}(t);\hat{\mathbf{S}},\mathcal{H}\big)\right\| \leq \left(\prod_{\ell=1}^{L}C_{\mathrm{H}_\ell}\right)\left\|\mathbf{X}_0-\hat{\mathbf{X}}_0\right\| = C_\Phi\left\|\mathbf{X}(t)-\hat{\mathbf{X}}(t)\right\|, \tag{C.11}$$

where the fact that the input to the GNN is the state at time t, i.e. $\mathbf{X}_0=\mathbf{X}(t)$. Finally, using (C.7) and (C.11) in (C.6), one obtains:

$$\left\|\mathbf{U}(t)-\hat{\mathbf{U}}(t)\right\| \leq \Gamma(\varepsilon)\Gamma_\Phi\left\|\mathbf{X}(t)\right\|+C_\Phi\|\mathbf{X}(t)-\hat{\mathbf{X}}(t)\|.$$

This simplifies (C.5) as

$$\begin{aligned}&\left\|\mathbf{B}\mathbf{U}(t-1)\bar{\mathbf{B}}-\hat{\mathbf{B}}\hat{\mathbf{U}}(t-1)\hat{\underline{\mathbf{B}}}\right\| \\ &\leq \Big(\|\mathbf{B}\|_2\|\bar{\mathbf{B}}-\hat{\underline{\mathbf{B}}}\|_\infty+\|\mathbf{B}-\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\Big)C_\Phi\|\mathbf{X}(t-1)\| \\ &\quad+\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\Gamma(\varepsilon)\Gamma_\Phi\left\|\mathbf{X}(t-1)\right\| \\ &\quad+\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty C_\Phi\|\mathbf{X}(t-1)-\hat{\mathbf{X}}(t-1)\|.\end{aligned} \tag{C.12}$$

Now, computing the size of the error signal in (C.1) and using the triangular inequality, together with (C.4) and (C.12), one obtains:

$$\begin{aligned}\left\|\mathbf{X}(t)-\hat{\mathbf{X}}(t)\right\| &\leq \Big(\|\hat{\mathbf{A}}\|_2\|\hat{\underline{\mathbf{A}}}\|_\infty+\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty C_\Phi\Big)\|\mathbf{X}(t-1)-\hat{\mathbf{X}}(t-1)\| \\ &\quad+\Big(\big(\|\mathbf{A}\|_2\|\bar{\mathbf{A}}-\hat{\underline{\mathbf{A}}}\|_\infty+\|\mathbf{A}-\hat{\mathbf{A}}\|_2\|\hat{\underline{\mathbf{A}}}\|_\infty\big) \\ &\quad+C_\Phi\big(\|\mathbf{B}\|_2\|\bar{\mathbf{B}}-\hat{\underline{\mathbf{B}}}\|_\infty+\|\mathbf{B}-\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\big)\Big)\|\mathbf{X}(t-1)\| \\ &\quad+\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\Gamma(\varepsilon)\Gamma_\Phi\left\|\mathbf{X}(t-1)\right\|.\end{aligned} \tag{C.13}$$

Recall that $\hat{\xi}=\|\hat{\mathbf{A}}\|_2\|\hat{\underline{\mathbf{A}}}\|_\infty+C_\Phi\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty$ and note that

$$\begin{aligned}&\big(\|\mathbf{A}\|_2\|\bar{\mathbf{A}}-\hat{\underline{\mathbf{A}}}\|_\infty+\|\mathbf{A}-\hat{\mathbf{A}}\|_2\|\hat{\underline{\mathbf{A}}}\|_\infty\big) \\ &+C_\Phi\big(\|\mathbf{B}\|_2\|\bar{\mathbf{B}}-\hat{\underline{\mathbf{B}}}\|_\infty+\|\mathbf{B}-\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\big)\leq\hat{C}_\xi\varepsilon\end{aligned} \tag{C.14}$$

for $\hat{C}_\xi$ as in (26). The value of $\|\mathbf{X}(t-1)\|$ can be further bounded as

$$\|\mathbf{X}(t-1)\| \leq \big(\|\mathbf{A}\|_2\|\bar{\mathbf{A}}\|_\infty+C_\Phi\|\mathbf{B}\|_2\|\bar{\mathbf{B}}\|_\infty\big)\|\mathbf{X}(t-2)\|. \tag{C.15}$$

Repeatedly applying this inequality, and noting that $\xi=\|\mathbf{A}\|_2\|\bar{\mathbf{A}}\|_\infty+C_\Phi\|\mathbf{B}\|_2\|\bar{\mathbf{B}}\|_\infty$, see (22), the bound on $\|\mathbf{X}(t-1)\|$ becomes

$$\|\mathbf{X}(t-1)\| \leq \xi^{t-1}\|\mathbf{X}(0)\|. \tag{C.16}$$

Using (C.14) and (C.16) back in (C.13), one obtains:

$$\begin{aligned}\left\|\mathbf{X}(t)-\hat{\mathbf{X}}(t)\right\| &\leq \hat{\xi}\ \|\mathbf{X}(t-1)-\hat{\mathbf{X}}(t-1)\|+\big(\hat{C}_\xi\varepsilon \\ &\quad+C_\Phi\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty\Gamma(\varepsilon)\Gamma_\Phi\big)\|\mathbf{X}(0)\|\xi^{t-1},\end{aligned} \tag{C.17}$$

which can be conveniently rewritten as

$$e_t \leq \hat{\xi}e_{t-1}+b\varepsilon\xi^{t-1}, \tag{C.18}$$

with

$$e_t=\|\mathbf{X}(t)-\hat{\mathbf{X}}(t)\|, \tag{C.19a}$$

$$\hat{\xi}=\|\hat{\mathbf{A}}\|_2\|\hat{\underline{\mathbf{A}}}\|_\infty+C_\Phi^L\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty, \tag{C.19b}$$

$$\xi=\|\mathbf{A}\|_2\|\bar{\mathbf{A}}\|_\infty+C_\Phi^L\|\mathbf{B}\|_2\|\bar{\mathbf{B}}\|_\infty, \tag{C.19c}$$

$$b=\big(\hat{C}_\xi+C_\Phi\|\hat{\mathbf{B}}\|_2\|\hat{\underline{\mathbf{B}}}\|_\infty(1+8\sqrt{N})\Gamma_\Phi\big)\|\mathbf{X}(0)\|, \tag{C.19d}$$

where the definition $\Gamma(\varepsilon)=(1+8\sqrt{N})\varepsilon$ was used to highlight the linearity with ε. By repeatedly applying (C.18), one arrives at:

$$e_t \leq b\varepsilon\sum_{\tau=0}^{t-1}\xi^{t-\tau-1}\hat{\xi}^\tau+\xi^t e_0. \tag{C.20}$$

Since the initial state of both the true system and the estimated one is the same, it holds that $e_0=\|\mathbf{X}(0)-\hat{\mathbf{X}}(0)\|=0$. Then, (C.20) becomes

$$e_t \leq b\varepsilon\sum_{\tau=0}^{t-1}\xi^{t-\tau-1}\hat{\xi}^\tau=\begin{cases}b\frac{\xi^t-\hat{\xi}^t}{\xi-\hat{\xi}} & \text{if } \xi\neq\hat{\xi}\\ bt\xi^{t-1} & \text{if } \xi=\hat{\xi}\end{cases}. \tag{C.21}$$

Now, recall that $|\xi^t-\hat{\xi}^t|\leq t\max\{\xi,\hat{\xi}\}^t|\xi-\hat{\xi}|$ so that (C.21) becomes $e_t\leq bt\max\{\xi,\hat{\xi}\}^{t-1}\varepsilon$. Finally, substituting the definitions of e_t as in (C.19a), $\hat{\xi}$ as in (C.19b), ξ as in (C.19c), and b as in (C.19d), we complete the proof. □

Now we prove Corollary 4 for the particular case when both systems $\mathcal{D}$ and $\hat{\mathcal{D}}$ are input-state stable.

Proof of Corollary 4.. From (28) in Theorem 3 it holds that $\hat{C}_t=t\max\{\xi,\hat{\xi}\}^{t-1}$. By assumption, it is known that $\xi<1$ and $\hat{\xi}<1$. Therefore, the function $t\max\{\xi,\hat{\xi}\}^{t-1}$ has a global maximum for $t\geq 0$. As a function of continuous $t\in\mathbb{R}$, this maximum is at $t=-1/\log(\max\{\xi,\hat{\xi}\})$ and gives the optimal value $-e^{-1}/(\max\{\xi,\hat{\xi}\}\times\log(\max\{\xi,\hat{\xi}\}))$. Thus, it holds that $\hat{C}_t\leq -e^{-1}\hat{C}_\Phi/(\max\{\xi,\hat{\xi}\}\times\log(\max\{\xi,\hat{\xi}\}))$, completing the first part of the proof. For the second part, note that, since $\xi<1$ and $\hat{\xi}<1$, then it holds that $\lim_{t\to\infty}t\max\{\xi,\hat{\xi}\}^{t-1}=0$. □

CRediT authorship contribution statement

Fernando Gama: Conceptualization, Writing – original draft, Investigation. **Somayeh Sojoudi:** Project administration, Validation, Visualization.

References

[1] F. Gama, S. Sojoudi, Graph neural networks for distributed linear-quadratic control, in: 3rd Annu. Conf. Learning Dynamics Control, Proc. Mach. Learning Res., Zürich, Switzerland, 2021.
[2] T. Kailath, Linear systems, Ser. Inform. Syst, Sci., Prentice-Hall, Englewood Cliffs, NJ, 1980.
[3] B.D.O. Anderson, J.B. Moore, Optimal Control: Linear Quadratic Methods, Ser. Inform. Syst, Sci., Prentice-Hall, Englewood Cliffs, NJ, 1989.
[4] S. Dean, H. Mania, N. Matni, B. Recht, S. Tu, On the sample complexity of the linear quadratic regulator, Found. Comput. Math. 20 (2020) 633–679.
[5] S. Fattahi, N. Matni, S. Sojoudi, Learning sparse dynamical systems from a single sample trajectory, in: 58th IEEE Conf. Decision, Control, IEEE, Nice, France, 2019, pp. 2683–2689.
[6] H.S. Witsenhausen, A counterexample in stochastic optimum control, SIAM J. Control 6 (1) (1968) 131–147.
[7] M. Rotkowitz, S. Lall, A characterization of convex problems in decentralized control, IEEE Trans. Autom. Control 51 (2) (2006) 274–286.
[8] S. Fattahi, G. Fazelnia, J. Lavaei, M. Arcak, Transformation of optimal centralized controllers into near-globally optimal static distributed controllers, IEEE Trans. Autom. Control 64 (1) (2019) 66–80.
[9] G. Fazelnia, R. Madani, A. Kalbat, J. Lavaei, Convex relaxation for optimal distributed control problems, IEEE Trans. Autom. Control 62 (1) (2017) 206–221.
[10] Y.-S. Wang, N. Matni, J.C. Doyle, A system-level approach to controller synthesis, IEEE Trans. Autom. Control 64 (10) (2019) 4079–4093.
[11] S. Fattahi, N. Matni, S. Sojoudi, Efficient learning of distributed linear-quadratic control policies, SIAM J. Control Optim. 58 (5) (2020) 2927–2951.
[12] F. Gama, E. Isufi, G. Leus, A. Ribeiro, Graphs, convolutions, and neural networks: from graph filters to graph neural networks, IEEE Signal Process. Mag. 37 (6) (2020) 128–138.
[13] F. Gama, J. Bruna, A. Ribeiro, Stability properties of graph neural networks, IEEE Trans. Signal Process. 68 (2020) 5680–5695.
[14] L. Ruiz, L.F.O. Chamon, A. Ribeiro, Graphon neural networks and the transferability of graph neural networks, in: 34th Conf. Neural Inform. Process. Syst., Neural Inform. Process. Syst. Foundation, Vancouver, BC, 2020, pp. 1702–1712.
[15] J.V. Capella, A. Bonastre, R. Ors, An advanced and distributed control architecture based on intelligent agents and neural networks, in: IEEE Int. Workshop Intell. Data Acquisition Advanced Computing Syst.: Technol. Appl., IEEE, Lviv, Ukraine, 2003, pp. 278–283.
[16] S.N. Huang, K.K. Tan, T.H. Lee, Decentralized control of a class of large-scale nonlinear systems using neural networks, Automatica 41 (9) (2005) 1645–1649.
[17] M.C. Choy, D. Srinivasan, R.L. Cheu, Neural networks for continuous online control, IEEE Trans. Neural Netw. 17 (6) (2006) 1511–1531.
[18] S.-Y. Chen, F.-J. Lin, Decentralized PID neural network control for five degree-of-freedom active magnetic bearing, Eng. Appl. Artific. Intell. 26 (3) (2013) 962–973.

[19] D. Liu, C. Li, H. Li, D. Wang, H. Ma, Neural-network-based decentralized control of continuous-time nonlinear interconnected systems with unknown dynamics, Neurocomputing 165 (2015) 90–98.
[20] S. Yang, Y. Cao, Z. Peng, G. Wen, K. Guo, Distributed formation control of nonholonomic autonomous vehicle via RBF neural network, Mech. Syst. Signal Process. 87 (B) (2017) 81–95.
[21] D. Wang, J. Qiao, L. Cheng, An approximate neuro-optimal solution of discounted guaranteed cost control design, IEEE Trans. Cybern. (2020), doi:10.1109/TCYB.2020.2977318. Early access
[22] D. Wang, M. Ha, J. Qiao, Data-driven iterative adaptive critic control toward an urban wastewater treatment plant, IEEE Trans. Ind. Electron. 68 (8) (2020) 7362–7369.
[23] F. Gama, Q. Li, E. Tolstaya, A. Prorok, A. Ribeiro, Decentralized control with graph neural networks, arXiv:2012.14906v3 [cs.LG] (2021) arXiv:2012.14906.
[24] J. Jahn, Introduction ot the Theory of Nonlinear Optimization, 3rd, Springer-Verlag, Berlin, Germany, 2007.
[25] A. Ortega, P. Frossard, J. Kovačević, J.M.F. Moura, P. Vandergheynst, Graph signal processing: overview, challenges and applications, Proc. IEEE 106 (5) (2018) 808–828.
[26] F. Gama, A. Ribeiro, Ergodicity in stationary graph processes: a weak law of large numbers, IEEE Trans. Signal Process. 67 (10) (2019) 2761–2774.
[27] S. Segarra, A. G. Marques, A. Ribeiro, Optimal graph-filter design and applications to distributed linear network operators, IEEE Trans. Signal Process. 65 (15) (2017) 4117–4131.
[28] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: 25th Conf. Neural Inform. Process. Syst., Neural Inform. Process. Syst. Foundation, Granada, Spain, 2011, pp. 2546–2554.
[29] V.N. Vapnik, The nature of statistical learning theory, Ser. Statist. Eng. Inform. Sci., 2nd, Springer-Verlag, New York, NY, 2000.
[30] D.P. Kingma, J.L. Ba, ADAM: a method for stochastic optimization, in: 3rd Int. Conf. Learning Representations, San Diego, CA, 2015, pp. 1–15.
[31] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[32] A. Nedić, Distributed gradient methods for convex machine learning problems in networks: distributed optimization, IEEE Signal Process. Mag. 37 (3) (2020) 92–101.
[33] O. Teke, P.P. Vaidyanathan, Random node-asynchronous updates on graphs, IEEE Trans. Signal Process. 67 (11) (2019) 2794–2809.
[34] M. Jin, J. Lavaei, Stability-certified reinforcement learning: acontrol-theoretic perspective, IEEE Access 8 (2020) 229086–229100.