Proceedings of Machine Learning Research vol XX:1–25, 2023

Compositional Neural Certificates for Networked Dynamical Systems

Songyuan Zhang SZHANG21@MIT.EDU
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA.

Yumeng Xiu YXIU2@ANDREW.CMU.EDU
Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.

Guannan Qu GQU@ANDREW.CMU.EDU
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.

Chuchu Fan CHUCHU@MIT.EDU
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA.

Abstract

Developing stable controllers for large-scale networked dynamical systems is crucial but has long been challenging due to two key obstacles: certifiability and scalablity. In this paper, we present a general framework to solve these challenges using compositional neural certificates based on ISS (Input-to-State Stability) Lyapunov functions. Specifically, we treat a large networked dynamical system as an interconnection of smaller subsystems and develop methods that can find each subsystem a decentralized controller and an ISS Lyapunov function; the latter can be collectively composed to prove global stability of the system. To ensure the scalability of our approach, we develop generalizable and robust ISS Lyapunov functions where a single function can be used across different subsystems and the certificates we produced for small systems can be generalized to be used on large systems with similar structures. We encode both ISS Lyapunov functions and controllers as neural networks, and propose a novel training methodology to handle the logic in ISS Lyapunov conditions that encodes the interconnection with neighboring subsystems. We demonstrate our approach in systems including Platoon, Drone formation control, and Power systems. Experimental results show that our framework can reduce the tracking error up to 75% compared with RL algorithms when applied to large scale networked systems.

Keywords: Neural Certificates, ISS Lyapunov Functions, Networked Dynamical Systems

1. Introduction

Large scale networked dynamical systems play an important role across a wide spectrum of real-world applications, including power grid (Zhao et al., 2014), vehicle platoons (Stankovic et al., 2000), drone swarms (Tedrake, 2022), transportation networks (Varaiya, 2013), etc. The control, and in particular stabilization, of such networked systems has long been recognized as a challenging problem as the dimension of the state and input spaces of such networked system is usually very high and existing methods often suffer from the "curse-of-dimensionality" (Powell, 2007).

Classical approaches for stabilization of dynamical systems include LQR (Linear Quadratic Regulator) for linear systems (Khalil et al., 1996; Dullerud and Paganini, 2013). For nonlinear systems, certificates like Lyapunov functions can be used to guide the search of a stabilizing controller and certify the stability of the closed-loop system (Slotine and Li, 1991). However, certificates are usually constructed on a case-by-case basis. While there exist approaches like SOS (Sum-Of-

Squares) that can construct certificate for general classes of nonlinear systems (Parrilo, 2000), they are not scalable to large-scale networked dynamical systems, since the number of polynomial coefficients in an SOS program grows exponentially w.r.t. the dimension of system (Parrilo, 2000).

A recent line of work parameterizes control certificates (*e.g.* Lyapunov functions, barrier functions) and controllers as neural networks (NNs) and learned them jointly from data (Chang et al., 2019; Jin et al., 2020; Chow et al., 2018). They have been successfully applied to nonlinear systems and achieved good performance on complex control tasks (Richards et al., 2018; Manek and Kolter, 2019) thanks to the representational power of NNs. However, the existing works mainly focus on single-agent systems with relatively small state space ($\leq$ 10 dimensions) or multi-agent systems without coupled dynamics (Chang et al., 2019; Qin et al., 2021b). Applying these approaches to large scale networked systems can be challenging due to the exponential growth of the sample complexity and the hardness of training NNs with large input spaces. Despite the challenge, many networked dynamics often contain sparse network structures that can be exploited to help training, and the question we try to answer in this paper is: *can we exploit network structure to learn neural certificates and stabilize large scale networked systems in a scalable and effective manner?*

To answer the question, we view the large networked dynamical system as a group of smaller subsystems interconnected through a graph. Instead of learning a single certificate for the entire system, we find a decentralized ISS (Input-to-State Stability) Lyapunov function (Sontag, 2013; Liu et al., 2011; Jiang and Liu, 2018) and a decentralized controller for each subsystem. Although ISS Lyapunov functions have been known for decades, it is not straightforward to adapt them as neural certificates for the stabilization of large networked systems due to the following reasons: 1) Existing ISS Lyapunov theory requires checking a condition involving global information of the networked system (Liu et al., 2011) thus is not entirely decentralized; 2) Existing ISS Lyapunov theory requires finding different ISS Lyapunov functions for each subsystem and therefore is computationally expensive for systems with many subsystems; 3) Each subsystem, as well as the corresponding ISS Lyapunov function, are intertwined with neighboring subsystems and therefore cannot be learned straightforwardly as Lyapunov functions for a single system such as those in Chang et al. (2019).

To tackle these challenges, we propose **Neur**al **ISS** Lyapunov funcions (NeurISS) that makes the following **contributions:** 1) We show that the ISS Lyapunov functions only need to satisfy a *local* condition involving local information from neighboring subsystems in order to collectively constitute a compositional certificate to certify the stability of the entire dynamical system (Lemma 1); 2a) We prove that under certain conditions, the compositional certificate for a small networked system can be generalized to be used on a larger system that has a similar structure without re-training (Lemma 2), which improves the scalability of the proposed approach as one can reduce a large training task to a smaller training task with a smaller network size; 2b) We extend the notion of ISS Lyapunov function to robust ISS Lyapunov function for control-affine systems (Lemma 3), so that similar subsystems that have different parameters can share the same ISS Lyapunov functions, which not only reduces the number of ISS Lyapunov functions we need to learn for large-scale networked systems but also also improves the robustness of the learned results against model uncertainties. 3) Furthermore, we develop a novel approach to encode the ISS logic condition that intertwines neighboring subsystems into the training loss function (Section 4).

We demonstrate NeurISS using three examples - Power systems, Platoon, and Drone formation control, and show that NeurISS can find certifiably stable controllers for networks of size up to 100 subsystems. Compared with centralized neural certificate approaches, NeurISS reaches similar results in small-scale systems, and can generalize to large scale systems that centralized approaches

cannot scale up to. Compared with LQR, NeurISS can deal with strong coupled networked systems like the microgrids, and reaches smaller tracking errors on both small and large scale systems. Compared with RL (PPO, LYPPO, MAPPO), our algorithm achieves similar or smaller tracking errors in small systems, and can hugely reduce the tracking errors in large systems (up to 75%).

Related Work. Safe machine learning, neural certificates, and reinforcement learning all have a rich literature. Due to space limit, we only mention the most related works.

Neural Certificates. Mostly related is the line of work on learning neural certificates. This line of work focuses on searching a controller together with a certificate that guarantees the soundness of the controller. Such neural certificates include Lyapunov-like functions for stability guarantees (Chang et al., 2019; Jin et al., 2020; Richards et al., 2018; Manek and Kolter, 2019; Abate et al., 2020; Dawson et al., 2021; Gaby et al., 2021), barrier functions for safety guarantees (Jin et al., 2020; Qin et al., 2021b; Xiao et al., 2021; Peruffo et al., 2021; Srinivasan et al., 2020), contraction metrics for tracking guarantees (Sun et al., 2020; Chou et al., 2021), etc. Through learning a proof of the correctness of the controllers, these approaches address the concerns about the safety, stability, and reliability of the controllers on a large variety of tasks, including precision quadrotor flight through turbulence (Sun et al., 2020), walking under model uncertainties (Castañeda et al., 2021), tracking with high-dimensional dynamics (Chou et al., 2021), and safe decentralized control of multi-agent systems (Qin et al., 2021b; Meng et al., 2021). Compared to these work, we learn ISS Lyapunov functions, which is decentralized and is scalable to large scale networked systems. We will compare the proposed approach with such neural certificate approach in Section 5.

ISS Lyapunov Function. The concept of ISS and ISS Lyapunov function is long established in control theory (Sontag, 2013). ISS Lyapunov function for networked dynamical systems was proposed in Jiang et al. (1996) for two subsystem case and generalized in Liu et al. (2011, 2012) for multiple subsystems (cf Jiang and Liu (2018) for a review). Compared to these works, our paper builds upon the ISS Lyapunov concept to learn neural certificates for networked systems.

Reinforcement Learning (RL). RL is a popular paradigm in the learning-to-control community with various approaches like (Deep) Q networks (Mnih et al., 2013) , policy optimization (Schulman et al., 2015, 2017) and multi-agent versions of them (Yu et al., 2021) (cf. Sutton and Barto (2018) for a review). However, standard RL is reward driven and does not formally guarantee stability. Recently, there has been research on learning certificates in the RL process (Berkenkamp et al., 2017; Chow et al., 2018; Cheng et al., 2019; Han et al., 2020; Chang and Gao, 2021; Zhao et al., 2021; Qin et al., 2021a), but none of them considers decentralized compositional certificates for networked systems. As a result, they are not scalable to large-scale networked systems as they lack the ability to deal with the sheer dimensions of the state space and the exponential grow of sample complexity. We will compare our approach with popular RL algorithms in Section 5.

2. Problem Setting

In this paper, we consider the following networked dynamical system involving n subsystems $\mathcal{N} = \{1, 2, \ldots, n\}$. The dynamics of each subsystem is given by

$$\dot{x}_i = f_i(x_i, x_{i_1}, x_{i_2}, \ldots, x_{i_{n_i}}, u_i) = f_i(x_i, x_{\mathcal{N}_i}, u_i) \tag{1}$$

where $x_i \in \mathbb{R}^{d_i}$, $u_i \in \mathbb{R}^{p_i}$ is the state and control inputs of each subsystem i, and $x_{\mathcal{N}_i} = (x_{i_1}, x_{i_2}, \ldots, x_{i_{n_i}})$ is used to denote the states of the neighbors of subsystem $i, \mathcal{N}_i = \{i_1, i_2, \ldots, i_{n_i}\}$ (not including i itself) which affect the dynamics of the subsystem i. We use $x = (x_1, \ldots, x_n)$,

$u = (u_1, \ldots, u_n)$, and $f = (f_1, \ldots, f_n)$ to denote the vector of states, actions and dynamics across all subsystems (*i.e.* the overall system), respectively. We also denote $d = \sum_{i=1}^{n} d_i$ and $p = \sum_{i=1}^{n} p_i$ to the dimension of x and u, respectively. Our goal is to design a controller $u = \pi(x)$ such that the closed-loop system is asymptotically stable around a goal set $\mathcal{X}^{\text{goal}}$, formally defined as follows.

Definition 1 *Consider a goal set $\mathcal{X}^{\text{goal}} := \mathcal{X}_1^{\text{goal}} \times \cdots \times \mathcal{X}_n^{\text{goal}}$ where each $\mathcal{X}_i^{\text{goal}}$ is a closed convex subset of $\mathbb{R}^{d_i}$. The closed-loop system is globally asymptotically stable about $\mathcal{X}_i^{\text{goal}}$ if for any initial state $x(0)$, the trajectory $x(t)$ satisfies $\lim_{t\to\infty} \text{dist}(x(t), \mathcal{X}^{\text{goal}}) = 0$, where $\text{dist}(x, \mathcal{X}^{\text{goal}}) := \inf_{y\in\mathcal{X}^{\text{goal}}} \|x - y\|$ is the distance between the point x and the set $\mathcal{X}^{\text{goal}}$.*

Example 1 *A simple networked system is the truck Platoon system with $n+2$ trucks, where the 0-th (leading) truck can drive freely within the speed and acceleration limits, and the $(n+1)$-th (last) truck will be driven in a way so that the total length of the platoon is roughly kept as a pre-defined constant. We assume other trucks are controllable but can only measure the distance to the two trucks directly in front of and behind itself. We want to control these trucks so that the trucks in the whole platoon spread evenly. Specifically, for truck $i \in \{1, 2, ..., n\}$, the states are given as $x_i = [p_i^f, p_i^b, v_i]^\top$ and the neighboring subsystems are trucks $\mathcal{N}_i = \{i-1, i+1\}$, where p_i^f is the distance between the i-th truck and the $(i-1)$-th truck, p_i^b is the distance between the i-th truck and the $(i+1)$-th truck, and v_i is the velocity of the i-th truck. The control input is the acceleration of the i-th truck. Therefore, the dynamics of truck i is $\dot{x}_i = [v_{i-1} - v_i, v_i - v_{i+1}, a_i]^\top$. The goal set for each truck i is uniquely defined as the set of states satisfying $p_i^f = p_i^b$.*

Notations. Function $\alpha : [0, \infty) \to [0, \infty)$ is said to be class-$\mathcal{K}$ if α is continuous, strictly increasing, and $\alpha(0) = 0$. Class-$\mathcal{K}$ function α is said to be class-$\mathcal{K}_\infty$ if $\lim_{a\to+\infty} \alpha(a) = +\infty$.

3. Compositional Neural Certificates

3.1. Decentralized Controller and ISS Lyapunov Functions for Networked Systems

Lyapunov functions are widely used to guarantee the stability of dynamical systems (see Appendix A in the full version of our paper Zhang et al. (2022) for an introduction). A common paradigm for stabilizing a dynamical system is to jointly search for a controller $u = \pi(x)$ and a Lyapunov function $V(x)$. However, this approach is not scalable for large scale networked dynamical systems due to the sheer dimension of the state space. In addition, another obstacle to scalability is the controller of the form $u = \pi(x)$, which requires global information of the entire network.

To address this issue, our framework NeurISS includes two key components: decentralized controllers and compositional certificates. We consider the class of *decentralized controllers* $u_i = \pi_i(x_i)$, which only needs local information within the small subsystem. Further, we consider *compositional certificates*, that is, instead of finding a single Lyapunov function $V(x)$ for the whole system, we find one Lyapunov function $V_i(x_i)$ for each subsystem i. The individual Lyapunov functions only depend on the subsystem state, which is much smaller in dimension. Further, based on Liu et al. (2011), we provide the following Lemma 1 which shows that when the individual Lyapunov functions V_i satisfy an ISS-style condition, they will certify the stability of the entire dynamical system. Since it is the collection of the ISS Lyapunov functions $\{V_i\}_{i=1}^n$ that certify the stability of the entire dynamics, we also call such ISS Lyapunov functions $\{V_i\}_{i=1}^n$ as a "compositional" certificate to distinguish them from typical certificates that only contain one Lyapunov function for the entire system. A proof of Lemma 1 is given in Appendix B in Zhang et al. (2022).

Lemma 1 *Suppose each subsystem has a decentralized controller $u_i = \pi_i(x_i)$ and a continuously differentiable function $V_i(x_i)$. Suppose the following is true: (1) For each i, there exist $\mathcal{K}_\infty$functions $\underline{\alpha}_i, \bar{\alpha}_i$ such that $\underline{\alpha}_i(\mathrm{dist}(x_i, \mathcal{X}_i^{\mathrm{goal}})) \le V_i(x_i) \le \bar{\alpha}_i(\mathrm{dist}(x_i, \mathcal{X}_i^{\mathrm{goal}}))$; (2) For each i, there exists $\alpha_i > 0$ and class-$\mathcal{K}$ functions $\chi_{ij}, j \in \mathcal{N}_i$ satisfying $\chi_{ij}(a) < a, \forall a > 0$, such that $\forall x_i, x_{\mathcal{N}_i}$,*

$$V_i(x_i) \ge \max_{j \in \mathcal{N}_i} \chi_{ij}(V_j(x_j)) \quad \Rightarrow \quad [\nabla V_i(x_i)]^\top f_i(x_i, x_{\mathcal{N}_i}, \pi_i(x_i)) \le -\alpha_i V_i(x_i). \tag{2}$$

Then, the closed-loop system under controllers $\pi_1, \ldots, \pi_n$ is globally asymptotically stable around $\mathcal{X}^{\mathrm{goal}}$. Such functions $V_i(x_i), i = 1, \ldots, n$ are called ISS Lyapunov functions.

We note that Lemma 1 is a variant of the result in Liu et al. (2011), in that we explicitly consider the network structure in the dynamics (1). As a result, in the ISS implication condition (2), we need to test $V_i(x_i)$ versus the max of $V_j(x_j)$ over only the neighbors $\mathcal{N}_i$, as opposed to the entire network as in Liu et al. (2011). This effectively makes (2) a condition that can be checked locally at each subsystem. One benefit of the local structure in the implication condition (2) is that it allows us to use certificates from smaller networks to compose certificates for larger networks that consist of blocks of the smaller networks. We will discuss this in detail in Section 3.2. Moreover, our results can also be robustified so a single ISS Lyapunov function can be used across different subsystems and handle uncertain parameters in the dynamics. We will explain this in detail in Section 3.3.

3.2. Network Generalizability

As discussed in Section 3.1, the condition (2) only involves f_i, V_i, and the Lyapunov functions of neighbors $\{V_j\}_{j \in \mathcal{N}_i}$. With such a local architecture, we present the following Lemma 2 that shows the decentralized controllers π_i and ISS Lyapunov functions V_i for a small system can be "ported over" to a larger dynamical system that has a similar symmetric structure to the smaller dynamical system. The proof of Lemma 2 is postponed to Appendix B in Zhang et al. (2022).

Lemma 2 *Consider a networked dynamical system with node set $\mathcal{N}$, neighborhood sets $\mathcal{N}_i$, and dynamics functions f_i, and suppose there exist decentralized controllers π_i such that the closed-loop dynamical system admits a compositional certificate V_i that satisfies the conditions in Lemma 1 with parameters χ_{ij}, α_i. Suppose there is another dynamical system with node set $\tilde{\mathcal{N}}$, neighborhood sets $\tilde{\mathcal{N}}_j$, and dynamics functions $\tilde{f}_i$. Suppose for each $j \in \tilde{\mathcal{N}}$, there exists a one-to-one map $\tau_j : \{j\} \cup \tilde{\mathcal{N}}_j \to \mathcal{N}$ such that $\tau_j(\tilde{\mathcal{N}}_j) = \mathcal{N}_{\tau_j(j)}$, and $\tilde{f}_j = f_{\tau_j(j)}$. Further, suppose $\forall j, j' \in \tilde{\mathcal{N}}$, $\forall \ell \in \tilde{\mathcal{N}}_j \cap \tilde{\mathcal{N}}_{j'}$, we have $V_{\tau_j(\ell)} = V_{\tau_{j'}(\ell)}$. Then, $\tilde{\pi}_j = \pi_{\tau_j(j)}$ is a stabilizing controller for the new system with compositional certificate $\tilde{V}_j = V_{\tau_j(j)}$.*

Example 2 *For the Platoon system, using Lemma 2, we can prove that a stabilizing controller for a 5-truck system $\pi_i, i = 1, \ldots, 5$ can be generalized to any system with $n > 5$. Let $\tilde{\mathcal{N}}$, $\tilde{\mathcal{N}}_i$, $\tilde{f}_i$ be the subsystems, neighborhood, and dynamics functions of the Platoon system with n trucks, and $\mathcal{N}$, $\mathcal{N}_i$, f_i be those of the system with 5 trucks. Note that in the Platoon system we have $\mathcal{N}_j = \{j-1, j+1\}$ and the same for $\tilde{\mathcal{N}}_j$. We define the one-to-one mapping τ_j in the following way: For the first truck, let $\tau_1(0) = 0, \tau_1(1) = 1, \tau_1(2) = 2$. For the last truck, let $\tau_n(n-1) = 4, \tau_n(n) = 5, \tau_n(n+1) = 6$. For other trucks $j = 2, 3, \ldots, n-1$, let $\tau_j(j-1) = 2, \tau_j(j) = 3, \tau_j(j+1) = 4$. Further, let $V_2 = V_3 = V_4$ and $\pi_2 = \pi_3 = \pi_4$ in the system with 5 trucks. In this way, we can check that $\forall j, j' \in \tilde{\mathcal{N}}$, $\forall \ell \in \tilde{\mathcal{N}}_j \cap \tilde{\mathcal{N}}_{j'}$, we have $V_{\tau_j(\ell)} = V_{\tau_{j'}(\ell)}$. Then following Lemma 2, we can conclude that $\tilde{\pi}_1 = \pi_1$, $\tilde{\pi}_n = \pi_5$, $\tilde{\pi}_j = \pi_2 = \pi_3 = \pi_4, j = 2, 3, \ldots, n-1$ are stabilizing controllers for the new system with certificates $\tilde{V}_1 = V_1, \tilde{V}_n = V_5$, and $\tilde{V}_j = V_2 = V_3 = V_4, j = 2, 3, \ldots, n-1$.*

3.3. Robust ISS Lyapunov

Many subsystems in a networked system are very similar in terms of dynamics and network structure but may have different parameters. Furthermore, system dynamics may have model uncertainties and unknown parameters (Dawson et al., 2021). For instance, in the Platoon example, trucks may have different weights, and the leading truck can have unknown velocity and acceleration, but the platoon system should be stabilized for any driving style of the leading truck. Having a robust version of ISS Lyapunov functions that can work for a set of different subsystems with different parameters can significantly reduce the number of ISS Lyapunov functions we need to find for a large networked system and also improve the robustness of the resulting controller. To tackle this, we show that the robust ISS Lyapunov functions can be established for control-affine systems taking the form: $\dot{x}_i = h_i(x_i, x_{\mathcal{N}_i}; \beta) + g_i(x_i, x_{\mathcal{N}_i}; \beta)u_i$, where $\beta \in \mathcal{B}$ is the parameter of the dynamics that models uncertainties. Such assumption is not restrictive and can cover a large range of physical systems, *e.g.* , systems following the manipulator function (Tedrake, 2022). Under this assumption, we further introduce robust ISS Lyapunov functions to guarantee the global asymptotic stability of the unknown system. The proof of Lemma 3 is postponed to the Appendix B in Zhang et al. (2022).

Lemma 3 (Robust ISS Lyapunov Functions) *Given a networked dynamical system with control-affine dynamics with bounded parametric uncertainty $\beta \in \mathcal{B}$, where $\mathcal{B}$ is the convex hull of parameters $\beta_1, \beta_2, \ldots, \beta_{n_\beta}$. If there exists ISS Lyapunov functions V_i satisfying the conditions in Lemma 1 for each $\beta_j, j \in \{1, 2, \ldots, n_\beta\}$, the dynamics h_i and g_i are affine with respect to β, then the closed-loop system is globally asymptotically stable with any $\beta \in \mathcal{B}$.*

Example 3 *The robust ISS Lyapunov functions can be directly used to the Platoon system. $v_0 = v_{n+1}$ is a parameter of this system, which is bounded between $v_{\min}$ and $v_{\max}$ by assumption. Following Lemma 3, if we can find ISS Lyapunov functions for both $v_{\min}$ and $v_{\max}$, we can ensure our system to be stable with any velocity of the leading truck.*

4. Learning Compositional Certificates and Controllers

Based on the compositional certificate developed in Section 3, we now focus on jointly learning the individual ISS Lyapunov functions and the decentralized controllers. We note that while there are many existing approaches on learning neural certificates (Dawson et al., 2021; Gaby et al., 2021), they can not be directly applied here because we have a unique imply condition (2) that can not be handled by the existing approaches. We will introduce a novel approach to incorporate the imply condition as specially designed loss terms. To proceed, we start with formally defining the parameterization of the decentralized controllers and the ISS Lyapunov functions.

Controller and ISS Lyapunov Functions Parameterization. We focus on decentralized controllers, in which the control u_i of subsystem i only depends on the subsystem state x_i, i.e. $u_i = \pi_i(x_i; \theta_i)$. Here π_i is an NN with θ_i as the parameters. We parameterize Lyapunov function $V_i(x_i)$ as $V_i(x_i; S_i, \omega_i, \nu_i) = x_i^\top S_i^\top S_i x_i + p_i(x_i; \omega_i)^\top p_i(x_i; \omega_i) + q_i(x_i; \nu_i)$. where $S_i \in \mathbb{R}^{d_i \times d_i}$ is a matrix of parameters, $p_i(x_i; \omega_i)$ is an NN with weights ω_i, and $q_i(x_i; \nu_i)$ is another NN with weights ν_i and ReLU as the output activation function, which is only applied to the output of the NN. The first term in $V_i(x_i)$ is a quadratic term to capture the linear part of the non-linear dynamics since Lyapunov functions for linear systems are often in quadratic forms. The second term is a sum-of-squares term to capture the polynomial part of the dynamics. Finally the third term is used to model the residues. Using this form, the ISS Lyapunov function satisfies $V_i(x_i) \geq 0$ by construction.

Sharing ISS Lyapunov Across Subsystems. Following Lemma 2 and Lemma 3, to reduce the number of neural networks and to make the ISS Lyapunov functions learned in small-scale networked system generalizable to large-scale systems, we use the following weight sharing technique. We let the subsystems share the same ISS Lyapunov functions if their dynamics are similar. In addition, we can also let the similar subsystems share the same controller. In this way, the number of trainable parameters can be reduced, and we can easily apply the controllers trained in small-scale systems to large-scale systems. For example, in the previous Platoon system, we can let the j-th truck, $j = 2, 3, \ldots, n-1$, share the same ISS Lyapunov functions and the same controller.

Gain Function Parameterizaton. We use linear functions to model the gain functions χ_{ij} in Equation (2). We let $\chi_{ij}(x) = \chi_i(x; k_i) = \text{Sigmoid}(k_i)x, \forall j$, where k_i is a trainable parameter, x is the scalar input of the gain functions, which is always the output of the ISS Lyapunov functions. In this way, the condition $\chi_{ij}(x) < x, \forall x > 0$ is satisfied by construction.

Loss Functions. A key challenge in learning ISS Lyapunov functions is how to ensure condition (2) is satisfied. We now propose a methodology that promotes (2). Let Boolean $A_i, B_i \in \{0, 1\}$ be[1]

$$A_i = V_i(x_i) \geq \max_{j \in \mathcal{N}_i} \chi_i(V_j(x_j)), \quad B_i = [\nabla V_i(x_i)]^\top f_i(x_i, x_{\mathcal{N}_i}, \pi_i(x_i)) \leq -\alpha_i V_i(x_i). \tag{3}$$

Then condition (2) can be written as $A_i \Rightarrow B_i$, which is the same as $\neg A_i \vee B_i$, or $\max\{\neg A_i, B_i\}$. However, this kind of formulation is not trainable for neural networks because Boolean variables are not differentiable. To settle this problem, we introduce the following losses:

$$\mathcal{L}_{A_i} = \text{ReLU}\left(V_i(x_i) - \max_{j \in \mathcal{N}_i} \chi_i(V_j(x_j)) + \epsilon_A\right), \tag{4}$$

$$\mathcal{L}_{B_i} = \text{ReLU}\Big([\nabla V_i(x_i)]^\top f_i(x_i, x_{\mathcal{N}_i}, \pi_i(x_i)) + \alpha_i V_i(x_i) + \epsilon_B\Big), \tag{5}$$

where ϵ_A and ϵ_B are small parameters that encourages strict satisfactions and generalization abilities (Dawson et al., 2021). $\mathcal{L}_{A_i}$ and $\mathcal{L}_{B_i}$ can address the problem introduced by Boolean variables A_i and B_i, but they introduce a new problem that $\max\{\neg A_i, B_i\}$ cannot be written as $\max\{\mathcal{L}_{A_i}, \mathcal{L}_{B_i}\}$ since the two losses are not comparable. To address this issue, we minimize the loss $\mu_{A_i}\mathcal{L}_{A_i} + \mu_{B_i}\mathcal{L}_{B_i}$ instead, where μ_{A_i} and μ_{B_i} are two hyper-parameters for balancing the two losses. Note that in practice, since we often simulate the dynamical systems in a discrete way, we can use two ways to calculate $\nabla V_i(x_i)$ in $\mathcal{L}_{B_i}$. First, we can directly calculate the gradient of $V_i(x_i)$ w.r.t. x_i. For the second method, we can just do an one-step simulation $x_i \to x_i^{\text{next}}$, and approximate $[\nabla V_i(x_i)]^\top f_i(x_i, x_{\mathcal{N}_i}, \pi_i(x_i))$ with $(V_i(x_i^{\text{next}}) - V_i(x_i))/\Delta t$, where Δt is the simulation time step.

To ensure the condition $V_i(x_i) = 0$ for $x_i \in \mathcal{X}_i^{\text{goal}}$ is satisfied, we introduce another loss term $\frac{1}{|\widehat{\mathcal{X}}_i^{\text{goal}}|}\sum_{x_i^{\text{goal}} \in \widehat{\mathcal{X}}_i^{\text{goal}}} |V_i(x_i^{\text{goal}})|$, where $\widehat{\mathcal{X}}_i^{\text{goal}}$ is a randomly sampled set of states from $\mathcal{X}_i^{\text{goal}}$. In addition, we add $\|\pi_i(x_i) - u_i^{\text{nominal}}\|^2$ to the loss, where u_i^{nominal} is the control signal calculated by some nominal controller. We use Droop controller and LQR controller in our experiments. We add the nominal controller so that the learned controller can explore the "informed region" near the nominal control signal rather than randomly, in order to accelerate the training. We do not need the nominal controller to be stable or optimal, and the learned controller behaves much better than the

1. For notational simplicity, from now on we omit all the notations of parameters in the function approximators.

nominal controller as shown in Section 5. The final loss function used in training is

$$\mathcal{L} = \sum_{i=1}^{n} \left[\frac{1}{|\widehat{\mathcal{X}}_i^{\text{goal}}|} \sum_{x_i^{\text{goal}} \in \widehat{\mathcal{X}}_i^{\text{goal}}} |V_i(x_i^{\text{goal}})| + \mu_{A_i} \mathcal{L}_{A_i} + \mu_{B_i} \mathcal{L}_{B_i} + \mu_{\text{ctrl}} \|\pi_i(x_i) - u_i^{\text{nominal}}\|^2 \right], \tag{6}$$

where μ_{ctrl} is a tuning parameter, and the training parameters are $\theta, S, \omega, \nu, k$.

Training Procedure. During training, we draw samples by randomly sampling states in the state space and in the goal set. We first initialize the controller by minimizing the loss $\|\pi_i(x_i) - u_i^{\text{nominal}}\|^2$, and then fix the controller to initialize the ISS Lyapunov function by minimizing the loss $\mathcal{L}_{B_i}$. After the initialization, we minimize loss (6) to train the controllers, the ISS Lyapunov functions, and the gain functions jointly. The contour plots of the learned robust ISS Lyapunov functions are provided in Appendix C.2.4 in Zhang et al. (2022).

5. Experiments

We demonstrate NeurISS in 3 environments including Power system, Platoon, and Drone, aiming to answer the following questions: How does NeurISS compare with other algorithms in the case of stabilizing networked systems? Can NeurISS perform similarly or surpass the centralized controllers in small-scale networked systems? Can NeurISS scale up to large-scale networked systems? We provide implementation details, introduction of the systems, and more results in the appendix.

Baselines. We compare NeurISS with both centralized and decentralized baselines. For centralized ones, we compare with the state-of-the-art RL algorithm PPO (Schulman et al., 2017), the RL-with-Lyapunov-critic algorithm LYPPO (Chang and Gao, 2021), and the centralized Neural CLF controller (NCLF) (Dawson et al., 2021). For decentralized ones, we compare with the classical LQR (Kwakernaak et al., 1974) controller and the multi-agent RL algorithm MAPPO (Yu et al., 2021). We hand-craft reward functions based on the common way of designing reward functions of tracking problems for the RL algorithms. For LQR, since the agents only have local observation, we calculate the goal point for the LQR controller based on local observations in each time step.

5.1. Environment Descriptions

Power Systems. We consider two control problems in power systems. Firstly, we consider a networked microgrid system introduced in Huang et al. (2021), where there is an interconnection of 5 microgrids. Each microgrid i has two states $x_i = (\delta_i, E_i)$ where δ_i is the voltage phase angle and E_i is the voltage magnitude. More details, including the dynamics, are given in Appendix C.1 in Zhang et al. (2022) and the goal is to design controllers so that δ_i, E_i can converge to their reference values $\delta_i^{\text{ref}}, E_i^{\text{ref}}$. Secondly, we consider a distribution grid voltage control problem (Shi et al., 2022) which we name GridVoltage8. The goal is to drive the distribution grid voltage to the nominal value 1.0. Due to space limits, the description of GridVoltage8 is deferred to Appendix C.1 in Zhang et al. (2022). Since the dynamics of the power systems are not separable, we use a droop controller as one of the baselines and the nominal controller for NeurISS and NCLF instead of LQR.

Platoon. The Platoon system has been introduced in the examples before. We use LQR controller as the nominal controller of NeurISS and NCLF. Because of the robustness and generalizability of NeurISS we let the controller and the ISS Lyapunov functions of the first and the n-th truck share the same weights, while other trucks also share the same weights. We let the controllers

Environment	Microgrid5	GridVoltage8	Platoon5	Drone2x2
NeurISS	2027.25 ± 18.15	$\mathbf{2483.70} \pm 1.68$	$\mathbf{2054.89} \pm 95.84$	$\mathbf{1713.73} \pm 13.35$
LQR	——	——	1894.64 ± 5.20	1209.15 ± 5.27
Droop	1431.30 ± 55.03	2251.70 ± 0.00	——	——
PPO	1970.89 ± 43.97	1086.72 ± 1.13	1489.32 ± 125.24	1489.32 ± 125.24
LYPPO	$\mathbf{2234.58} \pm 34.92$	1086.27 ± 2.12	707.08 ± 300.61	41.18 ± 6.05
MAPPO	1934.72 ± 149.66	1086.06 ± 0.27	1880.80 ± 155.49	1549.06 ± 271.32
NCLF	1887.36 ± 26.98	1078.19 ± 660.76	1979.02 ± 8.03	871.87 ± 46.02

Table 1: The expected reward of NeurISS and the baselines in the small-scale environments

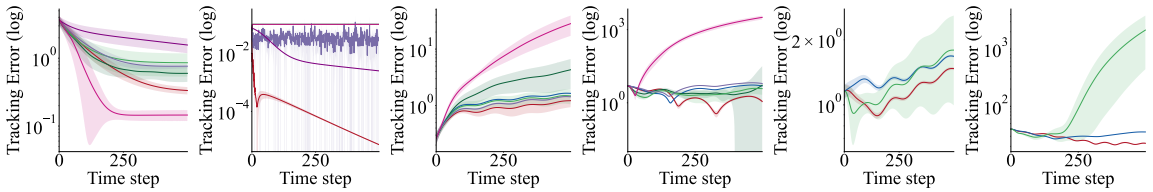


Figure 1: The tracking error in log scale w.r.t. time step of NeurISS and the baselines in the small-scale environments (*a-d*) and the large-scale environments (*e-f*). The line shows the mean tracking error while the shaded region shows the standard deviation.

of MAPPO share the weights in the same way for a fair comparison. In testing, we let the leading truck's acceleration follow a sin-like curve with clips, which is hard to track. In the small-scale training and testing, we use $n = 5$ trucks. In the large-scale testing, we use $n = 100$ trucks.

Drone. We design the Planar Drone Formation Control environment to further demonstrate the capability of NeurISS in complex networked systems. In this environment, in the beginning of the simulations, the planar drones (Tedrake, 2022) stay on the ground. As the simulations start, we want the drones to form a 2D mesh grid while tracking a given trajectory. The states of the drones are modeled as a 2-D platoon system, which is given by $x_i = [p_i^l, p_i^r, p_i^u, p_i^d, \theta_i, v_i^x, v_i^y, \omega_i]^\top$, where $p_i^l, p_i^r, p_i^u, p_i^d$ are the distances from drone i to the left, right, up, down drones, θ_i is the angle between the drone and the horizontal line, v_i^x and v_i^y are velocities and ω_i is the angular velocity. The control inputs of each drone are the forces generated by the two propellers. We use LQR controller as the nominal controller for NeurISS and NCLF, and let the controllers in NeurISS MAPPO share the same weights. Because of the robustness, we also let the ISS Lyapunov functions in NeurISS to share the same weights, so we only need 1 ISS Lyapunov function. In testing, we set the target trajectory to follow a horizontal line with a sin-like acceleration. In the small-scale training and testing, we use $2 \times 2 = 4$ drones. In the large-scale testing, we use $10 \times 10 = 100$ drones.

5.2. Results

The results show that compared with the baselines, NeurISS can achieve comparable or better rewards and tracking errors in small-scale environments, and significantly higher rewards and lower tracking errors in large-scale environments, which demonstrate its efficacy and generalizability.

Small-scale Experiments. In Table 1, we show the expected rewards and standard deviations of NeurISS and the baselines, and in Figure 1 (a-d) we show the tracking error w.r.t. the simulation time steps. We can observe that in small-scale system Microgrid5 (10 dimensions), NeurISS achieves the second highest expected reward, and second lowest tracking error, while LYPPO behaves the best. In GridVoltage8 (dimension 8) and larger systems Platoon5 and Drone2x2 (15 and 24 dimensions), NeurISS achieves the highest expected rewards and lowest tracking error. Note that in Platoon5 and Drone2x2, we do not have full knowledge of the tracking trajectory, and the trajectory changes fast, so the tracking error cannot converge to 0. NeurISS has this performance because of its ability to learn decentralized controllers *jointly* with the ISS Lyapunov functions as certificates. Compared with NCLF, NeurISS performs better because it is hard to find a global CLF for networked systems. PPO and MAPPO achieve lower reward than NeurISS.[2] They are policy gradient methods to approximate the solution of the Bellman equation, so there is no certificate of their stability. LYPPO, although outperforms NeurISS in very small systems (Microgrid), its performances drops a lot in larger systems (Platoon and Drone). This is because in small scale systems, with the guidance of CLF, RL can achieve the goal very quickly to maximize the cumulative reward, but NeurISS only seek to reach the goal without targeting on the convergence speed. Therefore, NeurISS convergences slower than LYPPO. However, in larger scale systems, because of the hardness of finding a correct global CLF, LYPPO receives the wrong guidance by the wrong CLF, and thus behaving much worse than NeurISS and even PPO and MAPPO. For the nominal controllers, LQR is designed for linear systems and the Droop controllers are hand-tuned. Therefore, their performance is hard to guarantee in complex nonlinear networked systems.

Large-scale Experiments. One key advantage of the proposed framework is network generalizability (Section 3.2), where the decentralized controllers and ISS Lyapunov functions trained in small networked systems can be directly applied to large networked systems without further training, while the centralized controllers needs a really long time to be trained on large-scale systems (Approximately 250 hours for Drone10x10). We test the 3 decentralized approaches, NeurISS, MAPPO, and LQR in large-scale Platoon and Drone systems, Platoon100 and Drone10x10, with 100 trucks and $10 \times 10 = 100$ drones. We show the tracking errors w.r.t. the simulation time steps in Figure 1 (e-f). We observe that NeurISS has the smallest tracking error in both environments, with large gaps to others, which show that NeurISS has the strongest scalability.

6. Conclusion

In this paper we propose a neural compositional certificate framework for stabilizing large scale networked dynamical systems. Limitations of the approach include: 1) the approach requires the knowledge of the dynamical system functions f_i; 2) the network generalizability result Lemma 2 requires a strong symmetric condition; 3) the robust ISS Lyapunov result Lemma 3 assumes the dynamical system is control affine; 4) the approach only learns a compositional certificate using finite samples but do not verify it, so in some sense the ISS Lyapunov functions we learn are only candidate ISS Lyapunov functions. Addressing these limitations are all interesting future directions.

2. For GridVoltage8, PPO, MAPPO, and LYPPO have almost identically bad rewards and tracking errors. This is because we use a professional solver PandaPower (Thurner et al., 2018) to simulate the underlying distribution grid. All three RL methods return controllers that quickly drive the system to an unsafe state, making the solver fail to solve the underlying model. In such cases, we simply set the reward and the tracking error assuming a 10% voltage deviation, which is a typical safety limit for real-world distribution grids (Shi et al., 2022).

References

Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Fernando Castañeda, Jason J Choi, Bike Zhang, Claire J Tomlin, and Koushil Sreenath. Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics. In *2021 American Control Conference (ACC)*, pages 3683–3690. IEEE, 2021.

Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809. IEEE, 2021.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov Control. In *Advances in Neural Information Processing Systems*, volume 32, pages 3245–3254, 2019.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3576–3583. IEEE, 2021.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 2018.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. *Conference on Robot Learning*, 2021.

Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*, volume 36. Springer Science & Business Media, 2013.

Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for lyapunov function approximation. *arXiv preprint arXiv:2109.13359*, 2021.

Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020.

Tong Huang, Sicun Gao, and Le Xie. A neural lyapunov approach to transient stability assessment of power electronics-interfaced networked microgrids. *IEEE Transactions on Smart Grid*, 13(1): 106–118, 2021.

Zhong-Ping Jiang and Tengfei Liu. Small-gain theory for stability and control of dynamical networks: A survey. *Annual Reviews in Control*, 46:58–79, 2018.

Zhong-Ping Jiang, Iven MY Mareels, and Yuan Wang. A lyapunov formulation of the nonlinear small-gain theorem for interconnected iss systems. *Automatica*, 32(8):1211–1215, 1996.

Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

IS Khalil, JC Doyle, and K Glover. *Robust and optimal control*. Prentice hall, 1996.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Huibert Kwakernaak, Raphael Sivan, and Bjor N D Tyreus. Linear optimal control systems. 1974.

Tengfei Liu, David J Hill, and Zhong-Ping Jiang. Lyapunov formulation of iss cyclic-small-gain in continuous-time dynamical networks. *Automatica*, 47(9):2088–2093, 2011.

Tengfei Liu, Zhong-Ping Jiang, and David J Hill. Lyapunov formulation of the iss cyclic-small-gain theorem for hybrid dynamical networks. *Nonlinear Analysis: Hybrid Systems*, 6(4):988–1001, 2012.

Steven H Low. Convex relaxation of optimal power flow—part i: Formulations and equivalence. *IEEE Transactions on Control of Network Systems*, 1(1):15–27, 2014.

Gaurav Manek and J Zico Kolter. Learning stable deep dynamics models. *Advances in neural information processing systems*, 2019.

Yue Meng, Zengyi Qin, and Chuchu Fan. Reactive and safe road user simulations using neural barrier certificate. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 370–388. Springer, 2021.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

Zengyi Qin, Yuxiao Chen, and Chuchu Fan. Density constrained reinforcement learning. In *International Conference on Machine Learning*, pages 8682–8692. PMLR, 2021a.

Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2021b.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yuanyuan Shi, Guannan Qu, Steven Low, Anima Anandkumar, and Adam Wierman. Stability constrained reinforcement learning for real-time voltage control. In *2022 American Control Conference (ACC)*, pages 2715–2721. IEEE, 2022.

Jean-Jacques E Slotine and Weiping Li. *Applied nonlinear control*. Prentice Hall, 1991.

Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Science & Business Media, 2013.

Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio A Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7139–7145. IEEE, 2020.

Srdjan S Stankovic, Milorad J Stanojevic, and Dragoslav D Siljak. Decentralized overlapping control of a platoon of vehicles. *IEEE Transactions on Control Systems Technology*, 8(5):816–832, 2000.

Dawei Sun, Susmit Jha, and Chuchu Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Russ Tedrake. *Underactuated Robotics*. 2022. URL http://underactuated.mit.edu.

L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, Nov 2018. ISSN 0885-8950. doi: 10.1109/TPWRS.2018.2829021.

Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

Wei Xiao, Ramin Hasani, Xiao Li, and Daniela Rus. Barriernet: A safety-guaranteed layer for neural networks. *arXiv preprint arXiv:2111.11277*, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Songyuan Zhang, Yumeng Xiu, Guannan Qu, and Chuchu Fan. Compositional neural certificates for networked dynamical systems. https://mit-realm.github.io/neuriss/static/full_version.pdf, 2022. Project page: https://mit-realm.github.io/neuriss.

Changhong Zhao, Ufuk Topcu, Na Li, and Steven Low. Design and stability of load-side primary frequency control in power systems. *IEEE Transactions on Automatic Control*, 59(5):1177–1189, 2014.

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

Appendix A. Lyapunov Function

Lyapunov functions are widely used to guarantee the stability of the dynamical systems. Lyapunov functions are formally defined in the following Proposition 1.

Proposition 1 *Given a dynamical system $\dot{x} = \bar{f}(x)$ where $\bar{f} : \mathbb{R}^d \to \mathbb{R}^d$, suppose there exists a differentiable and radially unbounded function $V : \mathbb{R}^d \to \mathbb{R}$ that satisfies the following conditions:*

$$V(x) = 0 \; \forall x \in \mathcal{X}^{\text{goal}}, \; V(x) > 0 \; \forall x \notin \mathcal{X}^{\text{goal}}, \text{ and } \nabla V(x)^\top \bar{f}(x) < 0 \; \forall x \notin \mathcal{X}^{\text{goal}}.$$

Then the system is asymptotically stable about $\mathcal{X}^{\text{goal}}$, and $V(x)$ is called a Lyapunov function.

A common paradigm for stabilizing a dynamical system is to jointly search for a controller $u = \pi(x)$ and a Lyapunov function $V(x)$ that satisfies the conditions in Proposition 1. However, finding a Lyapunov function for a large networked system is not trivial when the dimension of the state space (i.e. the input to the Lyapunov function) increases with the number of subsystems. In this paper, we utilize a decentralized compositional Lyapunov approach to improve the scalability of Lyapunov-based methods for large-scale system control.

Appendix B. Proofs

In this section, we provide the proof for the lemmas stated in the main content of the paper.

B.1. Proof of Lemma 1

Proof The proof follows similar steps as that in Liu et al. (2011) and we modify it to accomodate the network structure in our setting. Recall that we use $f(x, u)$ to denote the vector of the individual dynamical functions f_i. Further, we use $\pi(x)$ to denote $\pi(x) = [\pi_1(x_1), \ldots, \pi_n(x_n)]^\top$. With this notation, the closed-loop dynamical system can be written as

$$\dot{x} = f(x, \pi(x)).$$

Now consider the following function

$$V(x) = \max_{i \in \mathcal{N}} V_i(x_i).$$

Case 1. Suppose for a given x, $\max_{i\in\mathcal{N}} V_i(x_i)$ is uniquely achieved at $i^* \in \mathcal{N}$. Due to the continuous differentiability of the V_i's, we have $V(x') = V_{i^*}(x'_{i^*})$ for x' in a neighborhood of x, and as a result, $V(x)$ is continuously differentiable at x. Further, note that clearly $V_{i^*}(x_{i^*}) > \max_{j\in\mathcal{N}_{i^*}} V_j(x_j) \geq \max_{j\in\mathcal{N}_{i^*}} \chi_{i^*j}(V_j(x_j))$. Using the imply condition (**??**), we have,

$$\begin{aligned} &[\nabla V(x)]^\top f(x, \pi(x)) \\ &= [\nabla V_{i^*}(x_{i^*})]^\top f_{i^*}(x_{i^*}, x_{\mathcal{N}_{i^*}}, \pi_{i^*}(x_{i^*})) \\ &\leq -\alpha_{i^*} V_{i^*}(x_{i^*}) = -\alpha_{i^*} V(x). \end{aligned} \tag{7}$$

Case 2. Suppose for a given x, $\max_{i\in\mathcal{N}} V_i(x_i)$ is achieved at a set of multiple indices $\mathcal{I}$. Then, following a similar argument as Case 1 and using a continuity argument, there must exist a neighborhood $\mathcal{A}$ around x and constant $\lambda > 0$ such that for all $j \in \mathcal{I}$ and for all $y \in \mathcal{A}$,

$$[\nabla V_j(y_j)]^\top f_j(y_j, y_{\mathcal{N}_j}, \pi_j(y_j)) \leq -\lambda V_j(x_j), \tag{8}$$

and further, $\max_{i\in\mathcal{N}} V_i(y_i)$ must be achieved within $\mathcal{I}$.

Consider a trajectory of the system $\dot{y} = f(y, \pi(y))$ starting at $y(0) = x$. There must exist $\delta > 0$ s.t. $y(t) \in \mathcal{A}, \forall t \in [0, \delta]$. For any t within range $[0, \delta]$, suppose $\max_{i\in\mathcal{N}} V_i(y_i(t))$ is achieved at a certain $j \in \mathcal{I}$. Then, we have,

$$\begin{aligned}
&V(y(t)) - V(x) \\
&= V_j(y_j(t)) - V_j(y_j(0)) \\
&= \int_{\tau=0}^{t} \frac{d}{d\tau} V_j(y_j(\tau)) d\tau \\
&= \int_{\tau=0}^{t} [\nabla V_j(y_j(\tau))]^\top f_j(y_j(\tau), y_{\mathcal{N}_j}(\tau), \pi_j(y_j(\tau))) d\tau \\
&\leq -\lambda \int_{\tau=0}^{t} V_j(x_j) d\tau \\
&= -\lambda V_j(x_j) t = -\lambda V(x) t
\end{aligned}$$

where in the last inequality we have used (8). Therefore, whenever V is differentiable at x, we can divide the above by t and let $t \to 0$ to get

$$[\nabla V(x)]^\top f(x, \pi(x)) \leq -\lambda V(x). \tag{9}$$

Combining the above and (7), we have there exists a constant $\lambda' > 0$ s.t. $[\nabla V(x)]^\top f(x, \pi(x)) \leq -\lambda' V(x)$ whenever V is differentiable. Also, it is easy to check that V is continuously differentiable almost everywhere, and satisfies

$$\underline{\alpha}(\text{dist}(x, \mathcal{X}^{\text{goal}})) \leq V(x) \leq \bar{\alpha}(\text{dist}(x, \mathcal{X}^{\text{goal}})),$$

for $\mathcal{K}_\infty$ functions $\underline{\alpha}$ and $\bar{\alpha}$. Lastly, we can use the same argument as in the proof of Jiang et al. (1996, Thm 3.1) to show that the trajectory of the dynamical system must converge to the goal set. ■

B.2. Proof of Lemma 2

Proof Fix a $j \in \tilde{\mathcal{N}}$. For $j' \in \tilde{\mathcal{N}}_j$, we have

$$\tilde{V}_{j'} = V_{\tau_{j'}(j')} = V_{\tau_j(j')}. \tag{10}$$

Define $\tilde{\chi}_{jj'} = \chi_{\tau_j(j)\tau_j(j')}$ and $\tilde{\alpha}_j = \alpha_{\tau_j(j)}$. Suppose x_j and $x_{\tilde{\mathcal{N}}_j}$ are such that

$$\tilde{V}_j(x_j) \geq \max_{j'\in\tilde{\mathcal{N}}_j} \tilde{\chi}_{jj'}(\tilde{V}_{j'}(x_{j'})) \tag{11}$$

Using (10), we have (11) is equivalent to

$$V_{\tau_j(j)}(x_j) \geq \max_{j'\in\tilde{\mathcal{N}}_j} \chi_{\tau_j(j)\tau_j(j')}(V_{\tau_j(j')}(x_{j'})).$$

By (2), we have the above implies,

$$[\nabla V_{\tau_j(j)}(x_j)]^\top f_{\tau_j(j)}(x_j, x_{\tilde{N}_j}, \pi_{\tau_j(j)}(x_j)) \leq -\alpha_{\tau_j(j)} V_{\tau_j(j)}(x_j),$$

which is equivalent to

$$[\nabla \tilde{V}_j(x_j)]^\top \tilde{f}_j(x_j, x_{\tilde{N}_j}, \tilde{\pi}_j(x_j)) \leq -\tilde{\alpha}_j \tilde{V}_j(x_j). \tag{12}$$

This shows that (11) can imply (12). As such, $\tilde{V}_j$ is a valid compositional certificate for the new system. ■

B.3. Proof of Lemma 3

Proof Using the control-affine dynamics, we have

$$\begin{aligned} &[\nabla V_i(x_i)]^\top f_i(x_i, x_{\mathcal{N}_i}, \pi_i(x_i)) \\ &= [\nabla V_i(x_i)]^\top [h_i(x_i, x_{\mathcal{N}_i}; \beta) + g_i(x_i, x_{\mathcal{N}_i}; \beta)\pi(x_i)] \\ &= L_{h_i(\beta)} V_i(x_i) + L_{g_i(\beta)} V_i(x_i)\pi(x_i), \end{aligned} \tag{13}$$

where we denote $L_{h_i(\beta)} V_i(x_i) = [\nabla V_i(x_i)]^\top h_i(x_i, x_{\mathcal{N}_i}; \beta)$ and $L_{g_i(\beta)} V_i(x_i) = [\nabla V_i(x_i)]^\top g_i(x_i, x_{\mathcal{N}_i}; \beta)$ as the Lie derivatives of V_i along $h_i(x_i, x_{\mathcal{N}_i}; \beta)$ and $g_i(x_i, x_{\mathcal{N}_i}; \beta)$. By assumption, we have h_i and g_i are affine in β. In addition, the Lie derivatives $L_{h_i(\beta)} V_i$ and $L_{g_i(\beta)} V_i$ are affine in h_i and g_i, and Equation (13) is affine in $L_{h_i(\beta)} V_i$ and $L_{g_i(\beta)} V_i$. Therefore, the mapping from $\mathcal{B}$ to Equation (13) is affine which maps the convex hull of $\beta_1, \beta_2, \ldots, \beta_{n_\beta}$ to the convex hull of $L_{h_i(\beta_1)} V_i + L_{g_i(\beta_1)} V_i \pi_i, \ldots, L_{h_i(\beta_{n_\beta})} V_i + L_{g_i(\beta_{n_\beta})} V_i \pi_i$. As a result, if the conditions in Lemma 1 are satisfied for $\beta_i, i = 1, 2, \ldots, n_\beta$, then the conditions are satisfied for any $\beta \in \mathcal{B}$. Using Lemma 1, we can conclude that the closed-loop system is globally asymptotically stable with any $\beta \in \mathcal{B}$. ■

Appendix C. Experiment Details

Here we provide additional experimental details and results. We also provide the code of our experiments in the supplementary materials. Our experiments are run on a 64-core AMD 3990X CPU @ 3.60GHz and four NVIDIA RTX A4000 GPUs (one GPU for each training job).

C.1. Environment Details

C.1.1. POWER SYSTEMS

Networked Microgrid. We consider the networked microgrid system introduced by Huang et al. (2021), which is shown in Figure 2. In this environment, a power distribution network is divided into 5 regions (MG1, MG2, ..., MG5 in Figure 2). Each region i functions as a microgrid and two microgrids are neighbors when their corresponding regions are connected by a power line. Each

microgrid i has two states $x_i = (\delta_i, E_i)$ where δ_i means the voltage phase angle and E_i is the voltage magnitude. The dynamics of microgrid i is given by

$$M_{a,i}\dot{\delta}_i + (\delta_i - \delta_i^{\text{ref}}) = u_i^P + D_{a,i}(P_i^{\text{ref}} - G_{ii}E_i^2 - \sum_{j\in\mathcal{N}_i} E_i E_j Y_{ji}\cos(\delta_j - \delta_i - \sigma_{ji})), \tag{14a}$$

$$M_{v,i}\dot{E}_i + (E_i - E_i^{\text{ref}}) = u_i^Q + D_{v,i}(Q_i^{\text{ref}} + B_{ii}E_i^2 - \sum_{j\in\mathcal{N}_i} E_i E_j Y_{ji}\sin(\delta_j - \delta_i - \sigma_{ji})), \tag{14b}$$

where $M_{a,i}$ and $M_{v,i}$ are innertia coefficients, $D_{a,i}$ and $D_{v,i}$ are droop coefficients to provide internal droop controls, $\delta_i^{\text{ref}}, E_i^{\text{ref}}, P_i^{\text{ref}}, Q_i^{\text{ref}}$ are precomputed reference values, B_{ii}, G_{ii}, Y_{ji}, σ_{ji} are coefficients from the network admittance matrix. For a detailed description of the model, see Huang et al. (2021). Note that compared to Huang et al. (2021), we also introduce the control input $u_i = (u_i^P, u_i^Q)$ which represents the active and reactive power produced by the secondary control, and the goal is to design secondary controllers so that δ_i, E_i can converge to their reference values $\delta_i^{\text{ref}}, E_i^{\text{ref}}$.

During training, the training data are sampled from $\delta_i \in [-3, 3]$ and $E_i \in [-3, 3]$, and the data in the goal region are sampled from the region where $\delta_i = \delta_i^{\text{ref}}$ and $E_i = E_i^{\text{ref}}$. During testing, we set the simulation time interval $\Delta t = 0.01$, and randomly sample the initial states of the microgrids in $\delta_i \in [-2, 2]$ and $E_i \in [-3, 3]$. The number of simulation time steps is 500.

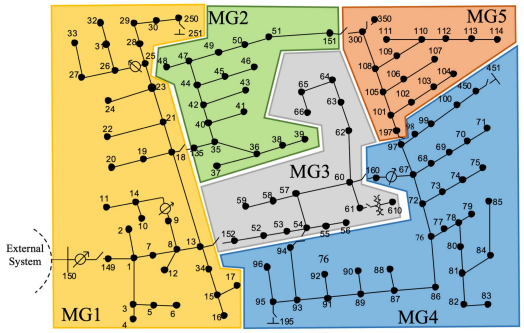


Figure 2: IEEE 123-node Test Feeder Huang et al. (2021)

Distribution grid voltage control. We consider a power system voltage control problem given in Shi et al. (2022). There is a power distribution network as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which consists of a set of nodes $\mathcal{N} = \{1, \ldots, n\}$ and edges $\mathcal{E}$. Each node $i \in \mathcal{N}$ is associated with a reactive power injection q_i, and a voltage magnitude v_i. We use q and v to denote the q_i, v_i stacked into a vector. The system dynamics is given as follows

$$\dot{q}(t) = u(t) = \pi(v(t)) \tag{15}$$

$$v(t) = v(q(t)) \tag{16}$$

where the state is the reactive power $q(t)$, and the control action is the change rate of $q(t)$. Critically, the voltage $v(t)$ is a function of the reactive power $q(t)$, denoted as $v(t) = v(q(t))$, and this function

is defined implicitly via the solution of a nonlinear algebraic equation system known as the power-flow equation (Low, 2014). In our experiment, we use PandaPower (Thurner et al., 2018) as the powerflow solver. The goal is to design a controller $u(t) = \pi(v(t))$ where the control action $u(t)$ depends on the voltage $v(t)$ such that in the close loop system, the voltage $v(t)$ across all nodes will converge to the nominal value (which is 1.0). In other words, the goal points of the problem consists of the set of reactive power such that the voltage is 1.0.

In our experiments, we consider a power distribution system that consists of 8 buses, see Figure 3. The buses are arranged in a line, each one is connected to a static generator. The nominal controller is similar to the droop control and is a proportion controller on the voltage deviation, $u_i(t) = c_i(v_i(t) - 1)$ (where c_i is a constant). This proportion controller is a standard controller used in practice.

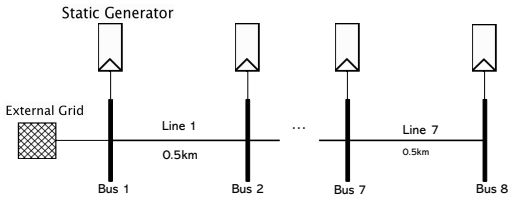


Figure 3: GridVoltage8

During training, the training data are sampled from $q_i \in [-0.001, 0.001]$, since Pandapower will give no solution when the variance of the training data is too large, in this case, we manually set the output voltage $v_i(t)$ to 0.9 for negative states $q_i(t)$ and 1.1 for positive ones. During testing, we set the simulation time interval $\Delta t = 0.01$, and sample the initial states of the power system from states of all 0. The number of simulation time steps is 500.

C.1.2. PLATOON

A platoon system is shown in Figure 4, which contains a list of trucks. The green truck is the leading truck (the 0-th truck), which can drive freely, and the orange truck is the last truck (the $(n+1)$-th truck). We want to control the trucks in the middle (the black trucks) so that the trucks in the whole truck platoon system spread evenly. As mentioned in the main pages, the state of the i-th truck is defined as $x_i = [p_i^f, p_i^b, v_i]^\top$, where p_i^f is the distance between the i-th truck and the $(i-1)$-th truck, p_i^b is the distance between the i-th truck and the $(i+1)$-th truck, and v_i is the velocity of the i-th truck. The state of the i-th truck is shown in Figure 4. For each relative position of the leading and last trucks, the goal set for each truck $i \in \{1, 2, ..., n\}$ is uniquely defined as the set of states satisfying $p_i^f = p_i^b$, which means that each middle truck aims to keep the distance to the front and behind trucks the same.

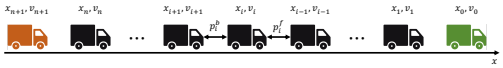


Figure 4: Platoon

During training, the training data are sampled from $p_i^f \in [0, 2]$, $p_i^b \in [0, 2]$, $v_i \in [0, 4]$, and the data in the goal region are sampled from the region where $p_i^f = p_i^b$. During testing, the leading truck follows a trajectory with initial velocity 2.0 and acceleration $\sin(5t\Delta t)$, where t is the simulation time step and $\Delta t = 0.01$ is the simulation time interval. Therefore, the velocity of the leading truck follows the profile shown in Figure 5. We randomly sample the controlled trucks' initial states in $p_i^f \in [0.6, 1.4]$, $p_i^b \in [0.6, 1.4]$, and $v_i \in [1.0, 1.2]$, so that the trucks need to speed up first to catch the leading truck. The number of simulation time steps is 500.

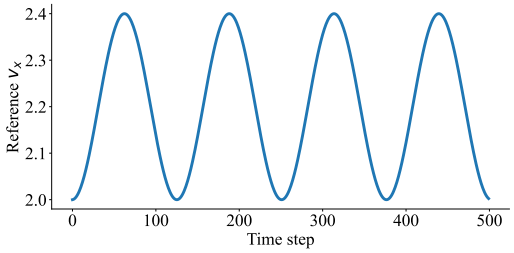


Figure 5: The velocity profile of the leading truck of the platoon system

C.1.3. PLANAR DRONE FORMATION CONTROL

A Planar Drone is shown in Figure 6 (Tedrake, 2022). For a single drone, the state is given by $x = [p_x, p_y, \theta, v_x, v_y, \omega]^\top$, where (p_x, p_y) is the position, (v_x, v_y) is the velocity, θ is shown in Figure 6, and ω is the changing rate of θ. The control inputs are forces generated by the two propellers u_1, u_2 shown in Figure 6. The dynamics of the drone is given by $\dot{x} = h(x) + g(x)u$, where

$$h(x) = \begin{bmatrix} v_x \\ v_y \\ \omega \\ 0 \\ -g \\ 0 \end{bmatrix}, \tag{17}$$

and

$$g(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{\sin\theta}{m} & -\frac{\sin\theta}{m} \\ \frac{\cos\theta}{m} & \frac{\cos\theta}{m} \\ \frac{r}{I} & -\frac{r}{I} \end{bmatrix}, \tag{18}$$

where m, I, r is the mass, moment of inertia, and the distance from the center to the base of the propeller, respectively.

For the formation control task, we design the states of the drones to be a 2-D platoon-like system, given by $x_i = [p_i^l, p_i^r, p_i^u, p_i^d, \theta_i, v_i^x, v_i^y, \omega_i]^\top$, where $p_i^l, p_i^r, p_i^u, p_i^d$ are the distances from drone i to the left, right, up, down drones. Other dimensions of the state are not coupled and are the same as the single drone system. The control inputs of each drone are the same as the single drone system.

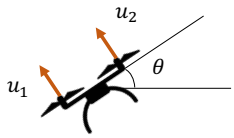


Figure 6: Planar Drone Dynamics (Tedrake, 2022)

Hyper-parameter	α_i	ϵ_A	ϵ_B	μ_{goal}	μ_{A_i}	μ_{B_i}	μ_{ctrl}
Microgrid	0.5	1.0	1.0	10	0.1	50.0	0.0
GridVoltage8	0.5	1.0	1.0	100	0.01	50.0	1.0
Platoon	1.0	1.0	1.0	100	0.1	50	0.001
PlanarDrone	0.2	1.0	1.0	100	0.01	3.0	0.2

Table 2: Hyper-parameters used during training NeurISS

During training, the training data are sampled from $p_i^l, p_i^r, p_i^u, p_i^d \in [0, 5]$, $\theta_i \in [-\pi/2, \pi/2]$, $v_i^x \in [-7, 7]$, $v_i^y \in [-5, 5]$, $\omega_i \in [-\pi/2, \pi/2]$, and the data in the goal region are sampled from the region where $p_i^l = p_i^r, p_i^u = p_i^d$, and $\theta_i = \omega_i = 0$. During testing, the trajectory that we want the drones to track follows the profile with initial velocity 1.0, and acceleration $0.5\sin(t\Delta t) - 0.25$, and the velocity is clipped below by 0.5. Here t is the simulation time step, and $\Delta t = 0.03$ is the simulation time interval. Therefore, the velocity of the tracking trajectory follows the profile shown in Figure 7. We randomly sample the drones' initial states in $p_i^l, p_i^r \in [0.8, 1.2]$, $p_i^u, p_i^d \in [0.09, 0.11]$, $v_i^x \in [0.85, 1.15]$, $v_i^y \in [-0.15, 0.15]$, $\theta_i \in [-0.05, 0.05]$, $\omega_i \in [-0.05, 0.05]$, so the drones need to first rise up and then follow the desired trajectory. The number of simulation time steps is 500.

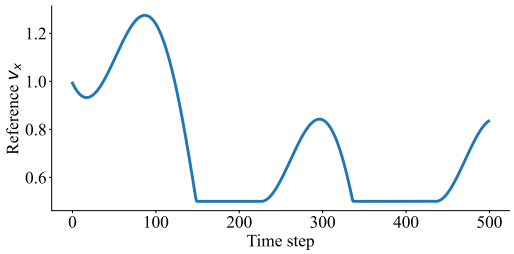


Figure 7: The velocity profile of the tracking trajectory of the planar drone formation control system

C.2. Implementation Details and Additional Results

In this section, we provide implementation details of NeurISS and the baselines, including the network structures, frameworks or the packages used, and the choice of the optimizer. We also provide the training details including the choice of the hyper-parameters, batch size, number of iterations, random seeds, and other mechanisms used in training. Moreover, we provide additional numerical results of the rewards of NeurISS and the baselines in large-scale environments, and the contour plots of the learned ISS Lyapunov functions that are not shown in the main pages due to the space limit.

Environment	Platoon100	PlanarDrone10x10
NeurISS	$\mathbf{1920.20} \pm 32.87$	$\mathbf{44723.37} \pm 39.58$
LQR	1812.63 ± 2.80	44184.47 ± 7.38
MAPPO	1816.53 ± 177.55	26594.20 ± 11563.69

Table 3: The expected reward of NeurISS and the baselines in the large scale environments

C.2.1. IMPLEMENTATION OF NEURISS

In our framework, there are two models to be trained: the neural ISS Lyapunov functions $V_i(x_i; S_i, \omega_i, \nu_i)$ and the controllers $\pi_i(x_i; \theta_i)$, where

$$\begin{aligned} V_i(x_i; S_i, \omega_i, \nu_i) = x_i^\top S_i^\top S_i x_i \\ + p_i(x_i; \omega_i)^\top p_i(x_i; \omega_i) + q_i(x_i; \nu_i), \end{aligned} \tag{19}$$

and $\pi_i(x_i; \theta_i)$ is a fully-connected multi-layer perceptron (MLP). $S_i \in \mathbb{R}^{d_i \times d_i}$ is a matrix of trainable parameters. The neural networks $p_i(x_i; \omega_i)$ and $\pi_i(x_i; \theta_i)$ are MLPs with two hidden layers with size 64 and $\tanh$ as the hidden activation function. $q_i(x_i; \nu_i)$ is an MLP with two hidden layers with size 64, $\tanh$ as the hidden activation function, and ReLU as the output activation function. To control the Lipschitz of the neural networks, we add the spectral normalization mechanism (Miyato et al., 2018) on every hidden layer of the neural networks. Our framework is implemented in PyTorch (Paszke et al., 2019) framework with ADAM (Kingma and Ba, 2014) as the optimizer.

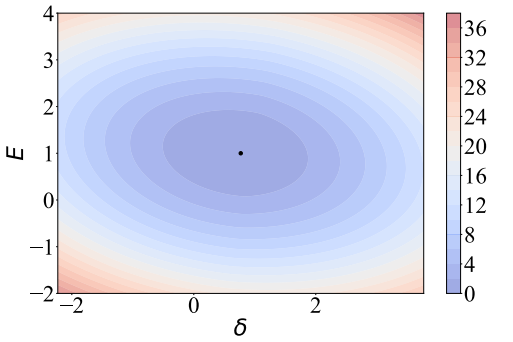
C.2.2. IMPLEMENTATION OF THE BASELINES

Our baselines include the droop controller, LQR (Kwakernaak et al., 1974), PPO (Schulman et al., 2017), LYPPO (Chang and Gao, 2021), MAPPO (Yu et al., 2021), and neural CLF (Dawson et al., 2021). The droop controller is only used in Microgrid5 and GridVoltage8. There is an internal droop controller given with the IEEE 123-node test feeder (Huang et al., 2021) so we directly use that controller as one of the baselines. The droop controller used in GridVoltage8 is similar to a proportion controller on the voltage deviation, $u_i(t) = c_i(v_i(t) - 1)$ (where c_i is a constant), this proportion controller is a standard controller used in practice. The LQR controller is used in the truck platoon system and the planar drone formation control system because these two systems are separable. We linearize the dynamics of the single truck and the single planar drone to calculate the LQR controller for them. PPO is implemented based on the open-source python package stablebaselines[3] (Raffin et al., 2021). LYPPO is implemented based on the official implementation. MAPPO is implemented based on the official implementation[4] (Yu et al., 2021). Neural CLF is implemented based on the official implementation[5] (Dawson et al., 2021).

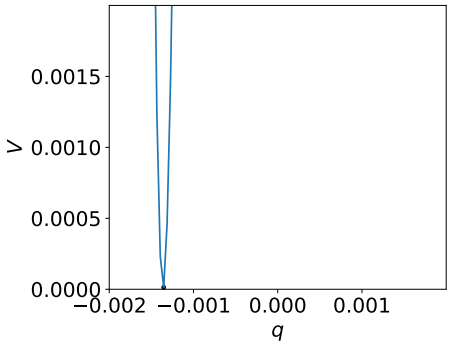
C.2.3. TRAINING DETAILS

In our framework, we include the hyper-parameters α_i, ϵ_A, ϵ_B, μ_{A_i}, μ_{B_i}, μ_{ctrl}. For simplicity, we omit the coefficient of the first term μ_{goal} in loss (6) in the main pages. We also include this

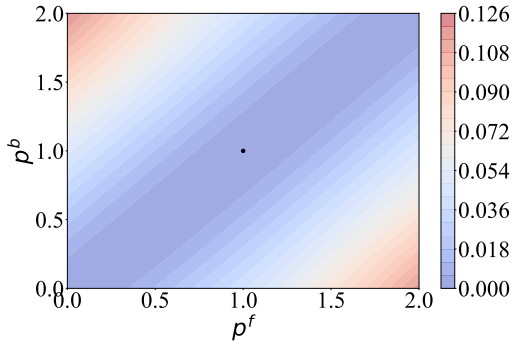
3. https://github.com/DLR-RM/stable-baselines3
4. https://github.com/marlbenchmark/on-policy
5. https://github.com/MIT-REALM/neural_clbf



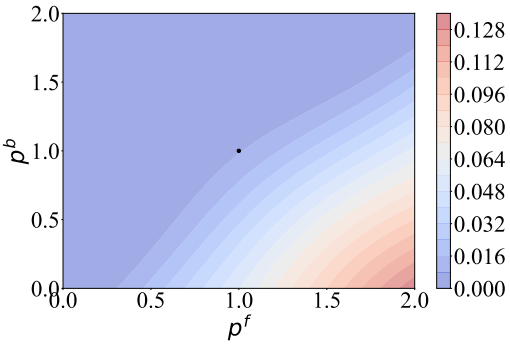
(*a*) Microgrid5



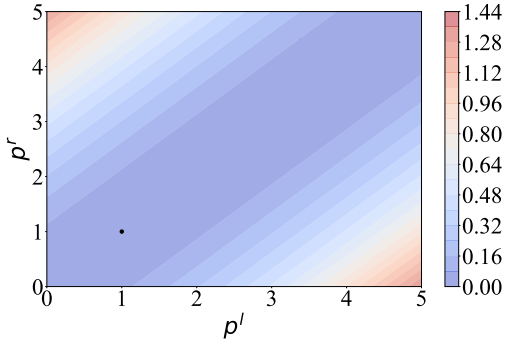
(*b*) GridVoltage8,$i = 1, ..., 8$



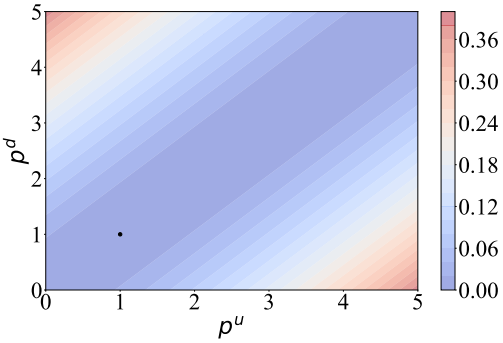
(*c*) Platoon5, $i = 1, 5$



(*d*) Platoon5, $i = 2, 3, 4$



(*e*) PlanarDrone2x2 (p_l v.s. p_r)



(*f*) PlanarDrone2x2 (p_u v.s. p_d)

Figure 8: Contour plots of the learned ISS Lyapunov functions, where the black dots show the goal point

hyper-parameter here. Note that omitting this coefficient in the main pages does not cause any problem because the coefficients μ_{goal}, μ_{A_i}, μ_{B_i}, μ_{ctrl} only controls the weight of each loss. If we set $\mu_{\text{goal}} = 1$ manually and divide other coefficients and the learning rate with μ_{goal}, we can get the same results. The exact values of the hyper-parameters are included in Table 2.

We further discuss the function of each hyper-parameter. α_i is the convergence rate of the ISS Lyapunov function. Larger α_i can make the closed-loop system converge faster to the goal, but it also makes the training harder. ϵ_A and ϵ_B are used to encourage strict satisfactions of loss (4) and (5), and to encourage the generalization abilities (Dawson et al., 2021). Larger ϵ_A and ϵ_B make the learned ISS Lyapunov functions have better generalization abilities, but also make the training harder. μ_{goal}, μ_{A_i}, μ_{B_i} are weights of different terms in the total loss (6). We choose them by balancing the value of each term to be at similar order of magnitudes. μ_{ctrl} controls the strength of the additional training signal of $\pi_i(x_i; \theta_i)$. If the nominal controller is good, we can use large μ_{ctrl} to accelerate the training, while if the nominal controller behaves badly, we use smaller μ_{ctrl} or even set $\mu_{\text{ctrl}} = 0$ to make sure that the nominal controller will not affect $\pi_i(x_i; \theta_i)$ too much.

During training, we set the batch size to be 2048, except for GridVoltage8 where the batch size is 1024, and train NeurISS for 10000 iterations. We set the learning rate to be 3×10^{-4} for the ISS Lyapunov functions $V_i(x_i; S_i, \omega_i, \nu_i$, 5×10^{-4} for the controllers $\pi(x_i; \theta_i)$, and 10^{-3} for the coefficients k_i for $\chi_i(x_i; k_i)$. To prevent overfitting, we add the weight decay mechanism with coefficient 10^{-3} for the ISS Lyapunov functions and the controllers. We train NeurISS and the baselines 4 times with random seeds $0, 1, 2, 3$.

C.2.4. ADDITIONAL RESULTS

In the main pages, we provide experimental results including comparison of the expected reward and the tracking error for small-scale environments, and the tracking error for large-scale environments. Here we provide more results of the experiments.

We provide the expected reward of NeurISS and the baselines in the large scale environments in Table 3. Note that to avoid negative rewards, we change the reward function in PlanarDrone10x10 to be $r = 100 - \sum_i (|p_i^l - p_i^r| + |p_i^u - p_i^d|)$. The reward function in all the environments is used to measure the cumulative tracking error, which is used to compare the converge speed of the tracking algorithms. We can observe that NeurISS achieves the highest reward in in both environments, which means NeurISS converges the fastest.

We provide the contour plots of the learned ISS Lyapunov functions in Figure 8. In the networked microgird environments, we train only one ISS Lyapunov function for all the subsystems because of the robustness. The contour plot of the learned ISS Lyapunov function is shown in Figure 8(*a*). We can observe that the learned ISS Lyapunov functions are in ellipse-like shapes indicating that the learned controller can go downhill w.r.t. the learned ISS Lyapunov functions and converge to the goal points (the black dots). In the networked GridVoltage8 environments, we train 8 ISS Lyapunov functions, one for each subsystem. The curve plots of the learned ISS Lyapunov function are shown in Figure 8(*b*). Since we have only one input state for each subsystem, the learned ISS Lyapunov functions are in quadratic-like shapes, and the learned controller can go downwards w.r.t. the learned ISS Lyapunov functions and converge to the goal points (the black dots). In the truck platoon environment, we train two ISS Lyapunov functions. One for the first and the last controllable truck ($i = 1$ and $i = n$), and another one for all other trucks ($i = 2, 3, \ldots, n-1$). The contour plots of the learned ISS Lyapunov functions are shown in Figure 8(*c*) and Figure 8(*d*). For trucks with $i = 1, n$, the learned ISS Lyapunov functions will make them converge to the line

where $p^f = p^b$. For trucks with $i = 2, 3, \ldots, n-1$, the learned ISS Lyapunov functions are sheer under the line $p^f = p^b$ and flat above that line. This is because the environment is parameterized by the velocity of the leading truck v_0, which is large compared with the initial velocity of other trucks. This causes that for the trucks in the middle, it is more often for them to reach the states where $p^f > p^b$, the region under $p^f = p^b$. Therefore, when the trucks are in states $p^f > p^b$, the sheer ISS Lyapunov functions will give them a strong signal to return to the line $p^f = p^b$. However when the trucks are in states $p^f < p^b$, they do not have to return to the $p^f = p^b$ quickly because the large v_0 can make them return automatically. In the drone formation control environment, we still train only one ISS Lyapunov function for all the subsystems because of the robustness. The learned ISS Lyapunov function is shown in Figure 8(*e*) and Figure 8(*f*), which can make the drone converge to the states where $p^l = p^r$ and $p^u = p^d$.