IEEE CSS IEEE Open Journal of Control Systems

Received 30 April 2024; revised 22 July 2024; accepted 31 July 2024. Date of publication 12 August 2024;
date of current version 25 September 2024. Recommended by Guest Editor Peter Seiler.

Digital Object Identifier 10.1109/OJCSYS.2024.3441768

Learning to Boost the Performance of Stable Nonlinear Systems

LUCA FURIERI (Member, IEEE), CLARA LUCÍA GALIMBERTI (Member, IEEE), AND GIANCARLO FERRARI-TRECATE (Senior Member, IEEE)

(Intersection of Machine Learning with Control)

École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

CORRESPONDING AUTHOR: LUCA FURIERI (e-mail: luca.furieri@epfl.ch).

This work was supported by the Swiss National Science Foundation (SNSF) under the NCCR Automation Grant Agreement 51NF40_80545. The work of Luca Furieri was supported by the SNSF for the Ambizione under Grant PZ00P2_208951.

ABSTRACT The growing scale and complexity of safety-critical control systems underscore the need to evolve current control architectures aiming for the unparalleled performances achievable through state-of-the-art optimization and machine learning algorithms. However, maintaining closed-loop stability while boosting the performance of nonlinear control systems using data-driven and deep-learning approaches stands as an important unsolved challenge. In this paper, we tackle the performance-boosting problem with closed-loop stability guarantees. Specifically, we establish a synergy between the Internal Model Control (IMC) principle for nonlinear systems and state-of-the-art unconstrained optimization approaches for learning stable dynamics. Our methods enable learning over specific classes of deep neural network performance-boosting controllers for stable nonlinear systems; crucially, we guarantee $\mathcal{L}_p$ closed-loop stability even if optimization is halted prematurely. When the ground-truth dynamics are uncertain, we learn over robustly stabilizing control policies. Our robustness result is tight, in the sense that all stabilizing policies are recovered as the $\mathcal{L}_p$ -gain of the model mismatch operator is reduced to zero. We discuss the implementation details of the proposed control schemes, including distributed ones, along with the corresponding optimization procedures, demonstrating the potential of freely shaping the cost functions through several numerical experiments.

INDEX TERMS Closed-loop stability, distributed control, internal model control, learning for control, optimal control, uncertain systems.

I. INTRODUCTION

The success of control systems across a broad spectrum of applications—from manufacturing to water, power, and transportation networks [1]—is rooted not only in advancements in sensing, computation, and communication but also in the growing availability of methods for designing model-based controllers capable of stabilizing nonlinear systems at nominal operating conditions.

However, in many applications, merely stabilizing the closed-loop system is not sufficient; achieving satisfactory performance is also crucial, often necessitating the integration of additional control loops. In Nonlinear Optimal Control (NOC), performance requirements are typically encoded in the shape of the cost function that the control policy strives to minimize. Consequently, it is beneficial to develop NOC algorithms that accommodate general nonlinear costs to enable sophisticated closed-loop behaviors, such as collision avoidance or waypoint tracking in swarms of robots.

In this paper, we tackle the following performance-boosting problem: given a discrete-time nonlinear system that is stable or has been pre-stabilized using a base controller, how can we enhance its performance during the transient—that is, before the system settles into a steady state—by employing general cost functions without compromising stability?

A first approach to designing performance-boosting regulators involves resorting to NOC methods with stability guarantees. Despite extensive research in this area [2], the problem is fully understood only when the system dynamics are linear and the cost admits a convex reformulation. For nonlinear systems, traditional methods for addressing NOC include dynamic programming and the maximum principle [3], [4]. However, the computation of NOC policies through these methods often faces significant computational challenges [4]. Furthermore, to ensure stability, stringent limitations must be imposed on the class of costs that can be utilized. An alternative approach to tackling performance-boosting is offered by receding-horizon control schemes, such as Nonlinear Model Predictive Control (NMPC) [5]. These controllers are based on real-time optimization; a finite-horizon NOC problem is solved at each time instant to determine the control input. However, a significant limitation of NMPC is that the control policy can seldom be precomputed and stored in an explicit form, which makes NMPC inapplicable when the control platform lacks the computational resources necessary to solve mathematical programs in real-time. Moreover, similar to NOC, ensuring stability requires imposing strong limitations on the class of admissible cost functions [5].

More recently, Reinforcement Learning (RL) and Deep Neural Networks (DNNs) have emerged as powerful tools that enable agents to understand and optimally interact with complex environments and dynamical systems, e.g., [6], [7]. Many RL approaches are based on minimizing arbitrary cost functions, calling for the use of broad sets of candidate nonlinear control policies. To this end, RL methods often employ families of policies that incorporate deep Neural Networks (NNs), due to their ability to model rich classes of nonlinear functions. These capabilities have led to remarkable applications, such as four-legged robots navigating challenging terrains [8] and drones that can outperform humans in races [9], [10]. On the other hand, general methodologies for designing RL policies for nonlinear dynamical systems, while ensuring closed-loop stability, are currently scarce and may be limited by strong assumptions [11], [12], [13]. As a result, so far the applicability of RL approaches has been mainly limited to systems that are not safety-critical.

Independent of their application in RL, NNs have been employed in model-based control since the 1990s for approximating nonlinear receding horizon policies [14], [15] or synthesizing nonlinear regulators from scratch [16]. Recent results on the design of provably stabilizing DNN control policies fall into two categories. The first one comprises constrained optimization approaches [11], [17], [18] that ensure global or local stability by enforcing Lyapunov-like inequalities during optimization. However, conservative stability constraints can severely restrict the range of admissible policies or fail to produce a viable controller even when it exists. Additionally, enforcing constraints such as linear matrix inequalities becomes a computational bottleneck in large-scale applications.

The second category embraces unconstrained optimization approaches, aiming to define classes of control policies with built-in stability guarantees [19], [20], [21]. These methods, which are similar to those developed in this paper, allow unconstrained optimization over finitely many parameters—using, for instance, standard gradient descent techniques—without sacrificing stability, regardless of the chosen parameter values. Optimizing over sets of stabilizing policies has two main benefits. First, it completely decouples the stabilization problem from the choice of the cost being optimized. Second, it enables *stability by design*, that is, the ability to guarantee closed-loop stability even if the policy optimization ends at a local minimum or is prematurely halted. However, these approaches are limited to discrete-time linear systems [19], [20] or to continuous-time systems in the port-Hamiltonian form [21]. While recent work surpasses the limitations above [22], [23], in real-world applications, the knowledge about the system model is not perfect. The impact of modeling errors on the parametrizations of stable closed-loop maps for nonlinear systems has remained largely unexplored.

A. CONTRIBUTIONS

This paper explores approaches to solve performance-boosting problems in general discrete-time, time-varying systems. Specifically, we develop unconstrained optimization approaches based on classes of state-feedback policies that induce closed-loop dynamics described by specific classes of stable and deep NNs.

After formally stating the performance-boosting problem in Section II, we present our first contribution, which provides a complete characterization of the class of stability-preserving controllers for stable systems. This result is presented in Section III and reveals that an Internal Model Control (IMC) structure [24], [25], [26] allows characterizing, without conservatism, the class of *all* stability-preserving controllers, where the only free parameter is an $\mathcal{L}_p$ operator. Our results hinge on adapting nonlinear variants of the Youla parametrization [27], [28], [29] to discrete-time systems in state-space with process noise, and revealing their connections with IMC schemes [24], [25], [26] in this setup.

Further, we examine the relationship with the recently proposed nonlinear System Level Synthesis (SLS) framework developed in [30]. In Section IV, our main contribution is that the proposed approach is compatible with scenarios where only an approximate system description is available, such as models identified from data or derived from simplified physical principles. Specifically, under a finite gain assumption on the model mismatch, stability can always be preserved by embedding a nominal system model and optimizing over nonlinear controllers with a sufficiently reduced gain on the free $\mathcal{L}_p$ parameter. Importantly, the method ensures vanishing conservatism in the class of parametrized stabilizing policies as the model uncertainty approaches zero. Additionally, by considering networks of interconnected subsystems, we demonstrate how the IMC structure of our controllers naturally lends itself to the development of distributed policies where the communication topology mirrors the subsystem couplings.

Finally, Section V bridges the gap between theoretical developments and computations, showing how to use Recurrent Equilibrium Networks (RENs) [31], [32] to obtain a finite-dimensional parametrization of performance-boosting controllers that can include DNNs. The final part of the paper in Section VI presents several simulations by considering coordination problems for mobile robots. Specifically, we show how, similarly to RL, the freedom in specifying the optimization cost allows designing NN controllers that can boost various forms of performance and safety, reaching beyond classical optimal control objectives consisting of the sum of stage-costs over time [3].

This paper builds upon our initial work [22] where we first derived the parametrization of all stabilizing controllers. However, unlike in [22], the IMC form of stabilizing controllers and the robustness analysis presented here are new. More specifically, the controllers in [22] were based on the nonlinear SLS parametrization introduced in [30], while the controllers in this paper rely on a much more intuitive IMC formulation. Additionally, the main technical contributions about robustness with vanishing conservatism included in this paper are novel and not included in [22]. Finally, the distributed control architectures and the majority of simulations presented in this work are not present in [22].

B. NOTATION

Signals and operators: The set of all sequences $\mathbf{x} = (x_0, x_1, x_2, \ldots)$, where $x_t \in \mathbb{R}^n$, $t \in \mathbb{N}$, is denoted as ℓ^n. Moreover, $\mathbf{x}$ belongs to $\ell_p^n \subset \ell^n$ with $p \in \mathbb{N} \cup \infty$ if $\|\mathbf{x}\|_p = (\sum_{t=0}^{\infty} |x_t|^p)^{\frac{1}{p}} < \infty$, where $|\cdot|$ denotes any vector norm. We say that $\mathbf{x} \in \ell_\infty^n$ if $\sup_t |x_t| < \infty$. When clear from the context, we omit the superscript n from ℓ^n and ℓ_p^n. An operator $\mathbf{A}$ is said to be ℓ_p-stable[1] if it is *causal* and $\mathbf{A}(\mathbf{w}) \in \ell_p^m$ for all $\mathbf{w} \in \ell_p^n$. Equivalently, we write $\mathbf{A} \in \mathcal{L}_p$. We say that an $\mathcal{L}_p$ operator $\mathbf{A} : \mathbf{w} \mapsto \mathbf{u}$ has finite $\mathcal{L}_p$-gain $\gamma(\mathbf{A}) > 0$ if $\|\mathbf{u}\|_p \leq \gamma(\mathbf{A}) \|\mathbf{w}\|_p$, for all $\mathbf{w} \in \ell_p^n$.

Time-series: We use the notation $x_{j:i}$ to refer to the truncation of $\mathbf{x}$ to the finite-dimensional vector $(x_i, x_{i+1}, \ldots, x_j)$. An operator $\mathbf{A} : \ell^n \to \ell^m$ is said to be *causal* if $\mathbf{A}(\mathbf{x}) = (A_0(x_0), A_1(x_{1:0}), \ldots, A_t(x_{t:0}), \ldots)$. If in addition $A_t(x_{t:0}) = A_t(x_{t-1:0}, 0)$, then $\mathbf{A}$ is said to be strictly causal. Similarly, we define $A_{j:i}(x_{j:0}) = (A_i(x_{i:0}), A_{i+1}(x_{i+1:0}), \ldots, A_j(x_{j:0}))$. For a matrix $M \in \mathbb{R}^{m \times n}$, $M\mathbf{x} = (Mx_0, Mx_1, \ldots) \in \ell^m$.

Graph theory: Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ described by the set of nodes $\mathcal{V} = \{1, \ldots, N\}$ and the set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, we denote set of neighbors of node i, including i itself by $\mathcal{N}_i = \{i\} \cup \{j \mid \{i, j\} \in \mathcal{E}\} \subseteq \mathcal{V}$. We denote with $\mathrm{col}_{j \in \mathcal{V}}(v^{[j]})$ a vector which consists of the stacked subvectors $v^{[j]}$ from $j = 1$ to $j = N$ and with $v^{[\mathcal{N}_i]}$ a vector composed by the stacked subvectors $v^{[j]}$ of all neighbors of node i, i.e., $v^{[\mathcal{N}_i]} = col_{j \in \mathcal{N}_i}(v^{[j]})$. For a signal $\mathbf{x} \in \ell^n$, where $x_t = col_{i \in \mathcal{V}}(x_t^{[i]})$, $x_t^{[i]} \in \mathbb{R}^{n_i}$, and $n = \sum_{i=1}^N n_i$, we denote with $\mathbf{x}^{[i]} \in \ell^{n_i}$ the sequence $\mathbf{x}^{[i]} = (x_0^{[i]}, x_1^{[i]}, \ldots)$. Similarly, we define $\mathbf{x}^{[\mathcal{N}_i]} = (x_0^{[\mathcal{N}_i]}, x_1^{[\mathcal{N}_i]}, \ldots)$.

II. THE PERFORMANCE-BOOSTING PROBLEM

We consider nonlinear discrete-time time-varying systems

$$x_t = f_t(x_{t-1:0}, u_{t-1:0}) + w_t \,, \quad t = 1, 2, \ldots \,, \tag{1}$$

where $x_t \in \mathbb{R}^n$ is the state vector, $u_t \in \mathbb{R}^m$ is the control input, $w_t \in \mathbb{R}^n$ stands for unknown process noise with $w_0 = x_0$, and $f_0 = 0$. The system model (1) is very general. For instance, it can describe the dynamics of the error between the state of a nonlinear system and a reference trajectory in ℓ_p. In *operator form*, system (1) is equivalent to

$$\mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \,, \tag{2}$$

where $\mathbf{F} : \ell^n \times \ell^m \to \ell^n$ is the strictly causal operator such that $\mathbf{F}(\mathbf{x}, \mathbf{u}) = (0, f_1(x_0, u_0), \ldots, f_t(x_{t-1:0}, u_{t-1:0}), \ldots)$. Note that $\mathbf{w} = (x_0, w_1, \ldots)$ and $\mathbf{u}$ collects all data needed for defining the system evolution over an infinite horizon. As an example, when the system (1) takes the Linear Time Invariant (LTI) form

$$x_t = Ax_{t-1} + Bu_{t-1} + w_t \,, \tag{3}$$

the model (2) becomes

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots \\ A & 0 & 0 & \cdots \\ 0 & A & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots \\ B & 0 & 0 & \cdots \\ 0 & B & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \end{bmatrix} + \begin{bmatrix} x_0 \\ w_1 \\ w_2 \\ \vdots \end{bmatrix} .$$

We consider disturbances with support $\mathcal{W}_t \subseteq \mathbb{R}^n$ following a random vector distribution $\mathcal{D}_t$, that is, $w_t \in \mathcal{W}_t$ and $w_t \sim \mathcal{D}_t$ for every $t = 0, 1, \ldots$. In order to control the behavior of system (1), we consider nonlinear, state-feedback, time-varying control policies

$$\mathbf{u} = \mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \ldots, K_t(x_{t:0}), \ldots) \,, \tag{4}$$

where $\mathbf{K} : \ell^n \to \ell^m$ is a *causal* operator to be designed. Note that the controller $\mathbf{K}$ can be dynamic, as K_t can depend on the whole past history of the system state. Since for each $\mathbf{w} \in \ell^n$ and $\mathbf{u} \in \ell^m$ the system (1) produces a unique state sequence $\mathbf{x} \in \ell^n$, (2) defines a unique transition operator

$$\mathcal{F} : (\mathbf{u}, \mathbf{w}) \mapsto \mathbf{x} \,,$$

which provides an input-to-state model of system (1). Similarly, for each $\mathbf{w} \in \ell^n$ the closed-loop system (1)–(4) produces unique trajectories. Hence, the closed-loop mapping $\mathbf{w} \mapsto (\mathbf{x}, \mathbf{u})$ is well-defined. Specifically, for a system $\mathbf{F}$ and a controller $\mathbf{K}$, we denote the corresponding induced closed-loop

[1] We also say that the operator is *stable*, for short, when the value of p is clear from the context.

operators $\mathbf{w} \mapsto \mathbf{x}$ and $\mathbf{w} \mapsto \mathbf{u}$ as $\mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{K}]$ and $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}]$, respectively. Therefore, we have $\mathbf{x} = \mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{K}](\mathbf{w})$ and $\mathbf{u} = \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}](\mathbf{w})$ for all $\mathbf{w} \in \ell^n$.

Definition 1: The closed-loop system (1)–(4) is ℓ_p-stable if $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}]$ and $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}]$ are in $\mathcal{L}_p$.

Our goal is to synthesize a control policy $\mathbf{K}$ solving the following problem.

Problem 1 (Performance boosting): Assume that $\boldsymbol{\mathcal{F}}$ lies in $\mathcal{L}_p$. Find $\mathbf{K}$ solving the finite-horizon Nonlinear Optimal Control (NOC) problem

$$\min_{\mathbf{K}(\cdot)} \quad \mathbb{E}_{w_{T:0}}\left[L(x_{T:0}, u_{T:0})\right] \tag{5a}$$

$$\begin{aligned} \text{s. t.} \quad & x_t = f_t(x_{t-1:0}, u_{t-1:0}) + w_t\,, \quad w_0 = x_0\,, \\ & u_t = K_t(x_{t:0})\,, \quad \forall t = 0, 1, \ldots\,, \\ & (\mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}]) \in \mathcal{L}_p\,, \end{aligned} \tag{5b}$$

where $L(\cdot)$ defines any piecewise differentiable lower bounded loss over realized trajectories $x_{T:0}$ and $u_{T:0}$, and the expectation $\mathbb{E}_{w_{T:0}}[\cdot]$ removes the effect of disturbances $w_{T:0}$ on the realized values of the loss.[2]

The main feature of (5) is that the cost is optimized over the finite horizon $0, \ldots, T$, but under the strict requirement that the closed-loop system is stable when it evolves over $0, \ldots, +\infty$. In other words, the feedback controller must preserve stability of $\boldsymbol{\mathcal{F}}$, and its role is to boost the performance of the system in the transient $0, \ldots, T$. As it will be clear in the sequel, we consider iterative control design algorithms based on gradient descent that exclusively search within sets of controllers that are stability-preserving by design. This guarantees closed-loop stability during the optimization of the policy parameters. Note also that, as it is standard in NOC, we do not expect gradient descent to find the globally optimal solution for any initialization – this is generally impossible for problems beyond Linear Quadratic Gaussian (LQG) control, which enjoy convexity of the cost and linearity of the optimal policies [33], [34]. Furthermore, the expected value in (5a) can seldom be computed[3] and is approximated by using samples of $w_{T:0}$. Our design guarantees that, in spite of all these limitations, closed-loop stability is never lost.

III. PARAMETRIZATION OF ALL STABILITY-PRESERVING CONTROLLERS

We show how to parametrize all and only the stability-preserving policies by using an IMC control architecture [24], [25], depending on an operator $\boldsymbol{\mathcal{M}}$ that can be freely chosen in $\mathcal{L}_p$. Specifically, the block diagram of the proposed control architecture is represented in Fig. 1 and it includes a copy of the system dynamics, which is used for computing the estimate $\widehat{\mathbf{w}}$ of the disturbance $\mathbf{w}$. A key advantage of the proposed IMC parametrization is its compatibility with recently proposed neural network dynamical system models such as those described in [31], [32]. As we discuss in Section V, these models enable the learning of performance-boosting stabilizing controllers by optimizing a set of free parameters $\theta \in \mathbb{R}^d$, for instance, through simple gradient descent. We are now in a position to introduce the main result.

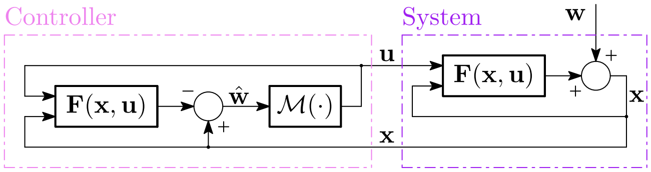

FIGURE 1. IMC architecture parametrizing of all stabilizing controllers in terms of one freely chosen operator $\mathcal{M} \in \mathcal{L}_p$.

Theorem 1: Assume that the operator $\boldsymbol{\mathcal{F}}$ is ℓ_p-stable, i.e. $\mathbf{x} \in \ell_p$ if $(\mathbf{w}, \mathbf{u}) \in \ell_p$, and consider the evolution of (2) where $\mathbf{u}$ is chosen as

$$\mathbf{u} = \boldsymbol{\mathcal{M}}(\mathbf{x} - \mathbf{F}(\mathbf{x}, \mathbf{u}))\,, \tag{6}$$

for a causal operator $\boldsymbol{\mathcal{M}} : \ell^n \to \ell^m$. Let $\mathbf{K}$ be the operator such that $\mathbf{u} = \mathbf{K}(\mathbf{x})$ is equivalent to (6).[4] The following two statements hold true.

1) If $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$, then the closed-loop system is ℓ_p-stable.
2) If there is a causal policy $\mathbf{C}$ such that $\mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{C}]$, $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{C}] \in \mathcal{L}_p$, then

$$\boldsymbol{\mathcal{M}} = \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{C}]\,, \tag{7}$$

gives $\mathbf{K} = \mathbf{C}$.

Proof: We prove 1). For compactness, define $\widehat{\mathbf{w}} = \mathbf{x} - \mathbf{F}(\mathbf{x}, \mathbf{u})$. As highlighted in [25], since there is no model mismatch between the plant $\boldsymbol{\mathcal{F}}$ and the model $\mathbf{F}$ used to define $\widehat{\mathbf{w}}$, one has $\widehat{\mathbf{w}} = \mathbf{w}$, hence opening the loop. More specifically, from Fig. 1 and (2) one has

$$\widehat{\mathbf{w}} = -\mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} = \mathbf{w}\,. \tag{8}$$

Therefore, by definition of the closed-loop maps, one has $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}] = \boldsymbol{\mathcal{M}}$ and $\mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{K}](\mathbf{w}) = \mathbf{F}(\mathbf{x}, \boldsymbol{\mathcal{M}}(\mathbf{w})) + \mathbf{w}$, $\forall \mathbf{w} \in \ell_p$. When $\mathbf{w} \in \ell_p$, one has $\mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}](\mathbf{w}) \in \ell_p$ because $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$. Moreover $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ and $\boldsymbol{\mathcal{F}} \in \mathcal{L}_p$ imply that the operator $\mathbf{w} \mapsto \mathbf{x}$ defined by the composition of the operators $\mathbf{w} \mapsto (\boldsymbol{\mathcal{M}}(\mathbf{w}), \mathbf{w})$ and $\boldsymbol{\mathcal{F}}$ is in $\mathcal{L}_p$ as well. This is due to the property that the composition of operators in $\mathcal{L}_p$ is in $\mathcal{L}_p$.

We prove 2). Set, for short, $\mathbf{\Psi}^{\mathbf{x}} = \mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{C}]$, $\mathbf{\Psi}^{\mathbf{u}} = \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{C}]$, $\mathbf{\Upsilon}^{\mathbf{x}} = \mathbf{\Phi}^{\mathbf{x}}[\mathbf{F}, \mathbf{K}]$, and $\mathbf{\Upsilon}^{\mathbf{u}} = \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F}, \mathbf{K}]$. By assumption, one has $\boldsymbol{\mathcal{M}} = \mathbf{\Psi}^{\mathbf{u}}$ and since $\mathbf{\Psi}^{\mathbf{u}} \in \mathcal{L}_p$ also $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$. By definition, $\mathbf{\Upsilon}^{\mathbf{u}}$ is the operator $\mathbf{w} \mapsto \mathbf{u}$ and, from (8) and Fig. 1, it coincides with $\boldsymbol{\mathcal{M}}$. Hence

$$\mathbf{\Psi}^{\mathbf{u}} = \mathbf{\Upsilon}^{\mathbf{u}}\,. \tag{9}$$

[2] Another common choice is to use $\max_{w_{T:0} \in \mathcal{W}_{T:0}}[\cdot]$ instead of the expectation. Other useful choices include $\mathrm{Var}_{w_{T:0}}[\cdot]$, $\mathrm{CVAR}_{w_{T:0}}[\cdot]$, and weighted combinations of all the above. In practice, one can approximate the chosen operator that removes the effect of disturbances from the cost by performing multiple experiments.

[3] For instance because it is too costly or the distribution $\mathcal{D}$ is unknown.

[4] This operator always exists because $\mathbf{F}(\mathbf{x}, \mathbf{u})$ is strictly causal. Hence u_t depends on the inputs $u_{t-1:0}$ and can be computed recursively from past inputs and $x_{t:0}$—see formula (11).

It remains to prove that $\mathbf{\Upsilon^x} = \mathbf{\Psi^x}$. Similar to [22], we proceed by induction. First, we show that $\Psi_0^x = \Upsilon_0^x$, where, as defined in Section I-B, Ψ_0^x and Υ_0^x are the components of $\mathbf{\Psi^x}$ and $\mathbf{\Upsilon^x}$ at time zero. Since $f_0 = 0$ and $w_0 = x_0$, one has from (1) that the closed-loop map $w_0 \mapsto x_0$ is the identity, irrespectively of the controller. Therefore $\Upsilon_0^x = \Psi_0^x = I$. Assume now that, for a positive $j \in \mathbb{N}$ we have $\Upsilon_i^x = \Psi_i^x$ for all $0 \leq i \leq j$. Since $(\mathbf{\Upsilon^x}, \mathbf{\Upsilon^u})$ and $(\mathbf{\Psi^x}, \mathbf{\Psi^u})$ are closed-loop maps, from (2) they verify

$$\Upsilon_{j+1}^x = F_{j+1}(\Upsilon_{j:0}^x, \Upsilon_{j:0}^u) + I, \; \Psi_{j+1}^x = F_{j+1}(\Psi_{j:0}^x, \Psi_{j:0}^u) + I. \tag{10}$$

But, from (9), one has $\Psi_{j:0}^u = \Upsilon_{j:0}^u$ and, by using the inductive assumption, one obtains $\Upsilon_{j+1}^x = \Psi_{j+1}^x$. This implies $\mathbf{K} = \mathbf{C}$. ■

Several comments are in order. First, Theorem 1 is about *nominal stability* only as there is no model mismatch between the plant model and the one used in the controller. We analyze robust stability in Section IV. Second, it is well known that many IMC architectures are sufficient for preserving stability, both in the linear [24] and the nonlinear [25] case.[5] It is also known that in the LTI setting, IMC is also necessary for preserving stability [35] and provides an alternative to the Youla-Koucera parametrization [36]. In this respect, Theorem 1 provides a necessary condition for preserving stability also for nonlinear systems. This result is perhaps not surprising given that necessary and sufficient conditions for stabilizing wide classes of input-output nonlinear models, in the spirit of the Youla- Koucera parametrization, have been derived since the 80's [27], [29]. However, these controllers are not conceived in the IMC form [24], [25], [26] and they consider actuation and measurement disturbances, while our setup allows for the presence of process noise.

The above insight is useful because the IMC structure facilitates the design and deployment of performance-boosting policies. First, IMC controllers are deployed using the block-diagram structure shown in Fig. 1. In equation form, for a chosen operator $\mathcal{M}$, one simply computes the control input as follows:

$$\widehat{w}_t = x_t - f_t(x_{t-1:0}, u_{t-1:0}), \tag{11a}$$

$$u_t = \mathcal{M}_t(\widehat{w}_{t:0}). \tag{11b}$$

Second, Theorem 1 highlights that it is sufficient to search in the space of operators $\mathcal{M} \in \mathcal{L}_p$ for describing all and only performance-boosting policies. While finding a parametrization of all operators $\mathcal{M} \in \mathcal{L}_p$ might be prohibitive, we will show in Section V that one can use NNs for describing broad subsets of these operators. Moreover, the IMC structure lends itself to the development of policies that enjoy a distributed structure (see Section IV).

1) THE CASE OF LTI SYSTEMS WITH NONLINEAR COSTS

Consider the linear system (3) and let z denote the forward time-shift operator. When the system is asymptotically stable, the classical Youla parametrization [36] states that all *linear state-feedback* stabilizing control policies $\mathbf{u} = \mathbf{Kx}$ can be written as

$$\mathbf{u} = \mathbf{Q}(z)\mathbf{x} - \frac{\mathbf{Q}(z)}{z}(A\mathbf{x} + B\mathbf{u}) \quad \mathbf{Q}(z) \in \mathcal{TF}_s, \tag{12}$$

where $\mathbf{Q}(z)$ is the so-called Youla parameter. Here, $\mathcal{TF}_s$ denotes the set of stable transfer matrices—that is, the set of matrices whose scalar entries are stable transfer functions. The class of linear control policies is globally optimal for standard LQG problems, and it allows optimizing over $\mathbf{Q} \in \mathcal{TF}_s$ using simple pole approximations and convex programming—we refer to [37], [38] for state-of-the-art results. However, nonlinear policies can be significantly more performing when the controller is distributed [39], or the cost function is nonlinear. As an immediate corollary of Theorem 1, and in accordance with the core contribution of [19] where the focus is on contracting closed-loops, we have the following result for linear systems controlled by nonlinear policies.

Corollary 1: Consider the linear system (3) and assume that it is asymptotically stable. Then, all and only control policies that make the closed-loop system ℓ_p-stable are expressed as

$$\mathbf{u} = \mathcal{M}\left(\mathbf{x} - \frac{(A\mathbf{x} + B\mathbf{u})}{z}\right), \tag{13}$$

where $\mathcal{M} \in \mathcal{L}_p$.

Proof: The proof follows from Theorem 1 upon realizing that the asymptotic stability of system (3) implies that the corresponding operator $\mathcal{F}$ is in $\mathcal{L}_p$, for any $p \geq 1$. ■

In conclusion, as expected, the linear Youla parametrization (12) is a special case of the proposed parametrization (13) with $\mathcal{M} = \mathbf{Q}$ and $\mathbf{Q} \in \mathcal{TF}_s$.

2) RELATIONSHIPS WITH [22] AND NONLINEAR SLS

In [22], we provided a slight generalization of Theorem 1 and the results in Section I by also considering unstable systems $\mathbf{x} = \tilde{\mathbf{F}}(\mathbf{x}, \mathbf{u}) + \mathbf{w}$ for which a pre-stabilizing controller $\mathbf{K}'$ exists, so that the overall policy is

$$\mathbf{u} = \mathbf{K}'(\mathbf{x}) + \mathcal{M}(\widehat{\mathbf{w}}). \tag{14}$$

By letting $\mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{F}(\mathbf{x}, \mathbf{K}'(\mathbf{x}) + \mathbf{u})$, and assuming that both $\mathcal{F}$ and $\mathbf{K}'$ lie in $\mathcal{L}_p$, Theorem 1 coincides with Theorem 2 in [22]. However, when $\mathbf{K}' \notin \mathcal{L}_p$, Theorem 2 in [22] highlights that $\mathcal{M} \in \mathcal{L}_p$ may no longer be a necessary condition for closed-loop ℓ_p-stability, while being still sufficient.

Moreover, as highlighted in [22], there is a deep link between Theorem 1 and the SLS parametrization of stabilizing controllers [30], [40]. The idea behind the SLS

[5] Note, however, that IMC in [25] is developed in terms of continuous-time nonlinear input-output models, for which the effect of process noise is difficult to analyze. Moreover, the control objective is to track a reference signal to the plant output, which raises the problem of approximating inverses of nonlinear operators. In our work, we use instead discrete-time input-to-state models and analyze the closed-loop maps from process noise to control inputs and system states. Moreover, our goal is to solve optimal control rather than tracking problems.

approach [30], [40] is to circumvent the difficulty of characterizing stabilizing controllers, by instead directly designing stable closed-loop maps. Let us define the set of all *achievable* closed-loop maps for system $\mathbf{F}$ as

$$\mathcal{CL}[\mathbf{F}] = \{(\mathbf{\Phi}^{\mathbf{x}}[\mathbf{F},\mathbf{K}],\, \mathbf{\Phi}^{\mathbf{u}}[\mathbf{F},\mathbf{K}]) \,|\, \mathbf{K} \text{ is causal}\}\,, \tag{15}$$

and the set of all *achievable and stable* closed-loop maps as

$$\mathcal{CL}_p[\mathbf{F}] = \{(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{CL}[\mathbf{F}] \,|\, (\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{L}_p\}\,. \tag{16}$$

Note that, if $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{CL}_p[\mathbf{F}]$, then $\mathbf{x} = \mathbf{\Psi}^{\mathbf{x}}(\mathbf{w}) \in \ell_p^n$ and $\mathbf{u} = \mathbf{\Psi}^{\mathbf{u}}(\mathbf{w}) \in \ell_p^m$ for all $\mathbf{w} \in \ell_p^n$. Based on Theorem III.3 of [30], and adding the requirement that the closed-loop maps must belong to $\mathcal{L}_p$, we summarize the main SLS result for nonlinear discrete-time systems.

Theorem 2 Nonlinear SLS parametrization [30]: The following two statements hold true.

1) The set $\mathcal{CL}_p[\mathbf{F}]$ of all achievable and stable closed-loop responses admits the following characterization:

$$\begin{aligned} \mathcal{CL}_p[\mathbf{F}] = \{(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) |\ & (\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \text{ are causal}\,, && (17a)\\ & \mathbf{\Psi}^{\mathbf{x}} = \mathbf{F}(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) + \mathbf{I}\,, && (17b)\\ & (\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{L}_p\}\,. && (17c) \end{aligned}$$

2) For any $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{CL}_p[\mathbf{F}]$, the operator $\mathbf{\Psi}^{\mathbf{x}}$ is invertible and the causal controller

$$\mathbf{u} = \mathbf{K}(\mathbf{x}) = \mathbf{\Psi}^{\mathbf{u}}\left((\mathbf{\Psi}^{\mathbf{x}})^{-1}(\mathbf{x})\right)\,, \tag{18}$$

is the only one that achieves the stable closed-loop responses $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}})$.

Theorem 2 clarifies that any policy $\mathbf{K}(\mathbf{x})$ achieving ℓ_p-stable closed-loop maps can be described in terms of two causal operators $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{L}_p$ complying with the nonlinear functional equality (17b). Therefore, the NOC problem admits an equivalent Nonlinear SLS (N-SLS) formulation:

$$\begin{aligned} \text{N-SLS:} \quad \min_{(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}})} \quad & \mathbb{E}_{w_{T:0}}\left[L(x_{T:0}, u_{T:0})\right] && (\star)\\ \text{s.t.} \quad & x_t = \Psi_t^x(w_{t:0})\,, \quad u_t = \Psi_t^u(w_{t:0})\,,\\ & (\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{CL}_p[\mathbf{F}]\,, t = 0, 1, \ldots \end{aligned}$$

According to Theorem 2, the constraint $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}}) \in \mathcal{CL}_p[\mathbf{F}]$ is equivalent to requiring that $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}})$ are causal and verify (17b)–(17c). The constraint (17b) simply defines the operator $\mathbf{\Psi}^{\mathbf{x}}$ in terms of $\mathbf{\Psi}^{\mathbf{u}}$ and it can be computed explicitly because $\mathbf{F}$ is strictly causal. The main challenge is to comply with (17c). Indeed, it is hard to generate $\mathbf{\Psi}^{\mathbf{u}} \in \mathcal{L}_p$ such that the corresponding $\mathbf{\Psi}^{\mathbf{x}}$ satisfies $\mathbf{\Psi}^{\mathbf{x}} \in \mathcal{L}_p$. The paper [30] suggests directly searching over ℓ_p-stable operators $(\mathbf{\Psi}^{\mathbf{x}},\mathbf{\Psi}^{\mathbf{u}})$ and abandoning the goal of complying with (17b) exactly. One can then study robust stability when (17b) only holds approximately as per Theorem IV.2 in [30]. However, with the exception of polynomial systems [41], this way of proceeding may result in conservative control policies or fail to produce a stabilizing controller. Instead, for the case of stable or pre-stabilized systems, Theorem 1 can be seen as a way of parametrizing all stabilizing controllers that circumvents completely the problem of fulfilling (17b)–(17c).

IV. BEYOND CLOSED-LOOP STABILITY: HANDLING MODEL UNCERTAINTY AND DISTRIBUTED ARCHITECTURES

This section tackles the performance boosting problem (Problem 1) under more intricate real-world constraints beyond just closed-loop stability. Firstly, Theorem 1 suffers from requiring perfect plant knowledge for controller design. In reality, ensuring closed-loop stability despite an imperfect model is crucial. Secondly, control policies in large-scale applications like power grids and traffic systems are inherently distributed. This means they rely solely on local sensor data and communication, posing significant challenges to achieving network-level robustness and stability.

A. ROBUSTNESS AGAINST MODEL-MISMATCH

Let us denote the nominal model available for design as $\widehat{\mathbf{F}}(\mathbf{x},\mathbf{u})$ and the real unknown plant as

$$\mathbf{F}(\mathbf{x},\mathbf{u}) = \widehat{\mathbf{F}}(\mathbf{x},\mathbf{u}) + \mathbf{\Delta}(\mathbf{x},\mathbf{u})\,, \tag{19}$$

where $\mathbf{\Delta}$ is a strictly causal operator representing the model mismatch. Let $\delta_t(x_{t-1:0}, u_{t-1:0})$ be the time representation of the mismatch operator $\mathbf{\Delta}$. Since for each sequence of disturbances $\mathbf{w} \in \ell^n$ and inputs $\mathbf{u} \in \ell^m$ the dynamics represented by (1) with $f_t(x_{t-1:0}, u_{t-1:0})$ replaced by $\widehat{f}_t(x_{t-1:0}, u_{t-1:0}) + \delta_t(x_{t-1:0}, u_{t-1:0})$ produces a unique state sequence $\mathbf{x} \in \ell^n$, the equation

$$\mathbf{x} = \mathbf{F}(\mathbf{x},\mathbf{u}) + \mathbf{w}\,, \tag{20}$$

defines again a unique transition operator $\mathcal{F}: (\mathbf{u},\mathbf{w}) \mapsto \mathbf{x}$, which provides an input-to-state model of the perturbed system.

Here, we show that when $\mathbf{\Delta}$ can be described by an $\mathcal{L}_p$ operator with finite gain, we can always design operators $\boldsymbol{\mathcal{M}}$ with sufficiently small $\mathcal{L}_p$-gain that stabilize the real closed-loop system. More specifically, letting $\gamma_{\mathbf{\Delta}}$ be the maximum $\mathcal{L}_p$-gain of the model mismatch $\mathbf{\Delta}$, it is possible to design controllers $\mathbf{K}$ that comply with the following robust version of the stability constraint (5b):

$$(\mathbf{\Phi}^{*}[\widehat{\mathbf{F}} + \mathbf{\Delta}, \mathbf{K}]) \in \mathcal{L}_p\,, \ * \in \{\mathbf{x},\mathbf{u}\}\,, \ \forall \mathbf{\Delta} |\, \gamma(\mathbf{\Delta}) \leq \gamma_{\mathbf{\Delta}}\,. \tag{21}$$

This result, which is given in the next theorem, refers to the control scheme in Fig. 2.

Theorem 3: Assume that the mismatch operator $\mathbf{\Delta}$ in (19) has finite $\mathcal{L}_p$-gain $\gamma(\mathbf{\Delta})$. Furthermore, assume that the operator $\mathcal{F}$ has finite $\mathcal{L}_p$-gain $\gamma(\mathcal{F})$. Then, for any $\boldsymbol{\mathcal{M}}$ such that

$$\gamma(\boldsymbol{\mathcal{M}}) < \gamma(\mathbf{\Delta})^{-1}(\gamma(\mathcal{F}) + 1)^{-1}\,, \tag{22}$$

the control policy given by

$$\begin{aligned} \widehat{w}_t &= x_t - \widehat{f}_t(x_{t-1:0}, u_{t-1:0})\,, && (23a)\\ u_t &= \mathcal{M}_t(\widehat{w}_{t:0})\,, && (23b) \end{aligned}$$

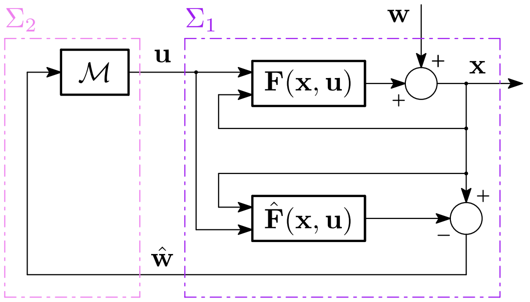
Σ_2
Σ_1
$\mathbf{w}$
$\mathbf{u}$
$\boldsymbol{\mathcal{M}}$
$\mathbf{F}(\mathbf{x},\mathbf{u})$
$+$
$+$
$\mathbf{x}$
$\hat{\mathbf{F}}(\mathbf{x},\mathbf{u})$
$+$
$-$
$\hat{\mathbf{w}}$

FIGURE 2. **The closed-loop system when the nominal model $\widehat{\mathbf{F}}(\mathbf{x},\mathbf{u})$ used in the IMC controller and the real plant $\mathbf{F}(\mathbf{x},\mathbf{u}) = \widehat{\mathbf{F}}(\mathbf{x},\mathbf{u}) + \boldsymbol{\Delta}(\mathbf{x},\mathbf{u})$ differ by the perturbation $\boldsymbol{\Delta} \in \mathcal{L}_p$. Compared to Fig. 1 the blocks have been rearranged to highlight the subsystems used in the small-gain argument adopted in the proof of Theorem 3.**

stabilizes the closed-loop system.

Proof: We first show that operators $\mathbf{F}$ and $\mathcal{F}$ verify

$$\mathbf{F}(\mathcal{F}(\mathbf{u},\mathbf{w}),\mathbf{u}) = \mathcal{F}(\mathbf{u},\mathbf{w}) - \mathbf{w}\,. \tag{24}$$

This follows by substituting $\mathbf{x} = \mathcal{F}(\mathbf{u},\mathbf{w})$ in (20). We now compute the $\mathcal{L}_p$-gain of the operator $\Sigma_1 : (\mathbf{u},\mathbf{w}) \mapsto \widehat{\mathbf{w}}$ in the right frame of Fig. 2:

$$\begin{aligned}\widehat{\mathbf{w}} &= \mathcal{F}(\mathbf{u},\mathbf{w}) - \widehat{\mathbf{F}}(\mathcal{F}(\mathbf{u},\mathbf{w}),\mathbf{u})\\ &= \mathbf{F}(\mathcal{F}(\mathbf{u},\mathbf{w}),\mathbf{u}) - \widehat{\mathbf{F}}(\mathcal{F}(\mathbf{u},\mathbf{w}),\mathbf{u}) + \mathbf{w}\\ &= \boldsymbol{\Delta}(\mathcal{F}(\mathbf{u},\mathbf{w}),\mathbf{u}) + \mathbf{w}\,, \end{aligned}\tag{25}$$

where the first equality follows from (24). Using the definition of $\mathcal{L}_p$-gain for the operator $\mathbf{y} = \boldsymbol{\Delta}(\mathbf{x},\mathbf{u})$ one has $||\mathbf{y}||_p \leq \gamma(\boldsymbol{\Delta})(||\mathbf{x}||_p + ||\mathbf{u}||_p)$, and, by using (25) and $\mathbf{u} = \boldsymbol{\mathcal{M}}(\widehat{\mathbf{w}})$, one obtains[6]

$$\begin{aligned}&||\widehat{\mathbf{w}}|| \leq \gamma(\boldsymbol{\Delta})(||\mathcal{F}(\mathbf{u},\mathbf{w})|| + ||\mathbf{u}||) + ||\mathbf{w}||\\ &\leq \gamma(\boldsymbol{\Delta})(\gamma(\mathcal{F})||\mathbf{w}|| + \gamma(\mathcal{F})||\mathbf{u}|| + ||\mathbf{u}||) + ||\mathbf{w}||\\ &\leq (\gamma(\boldsymbol{\Delta})\gamma(\mathcal{F}) + 1)||\mathbf{w}|| + \gamma(\boldsymbol{\Delta})(\gamma(\mathcal{F}) + 1)\gamma(\boldsymbol{\mathcal{M}})||\widehat{\mathbf{w}}||\,.\end{aligned}$$

By gathering all the terms involving $||\widehat{\mathbf{w}}||$ to the left-hand side we obtain

$$(1 - \gamma(\boldsymbol{\Delta})\gamma(\boldsymbol{\mathcal{M}})\,(\gamma(\mathcal{F})+1))||\widehat{\mathbf{w}}|| \leq (\gamma(\boldsymbol{\Delta})\gamma(\mathcal{F})+1)||\mathbf{w}||\,.$$

Since (22) holds, we have that $1 - \gamma(\boldsymbol{\Delta})\gamma(\boldsymbol{\mathcal{M}})(\gamma(\mathcal{F}) + 1) > 0$, and hence

$$||\widehat{\mathbf{w}}|| \leq \left(\frac{\gamma(\boldsymbol{\Delta})\gamma(\mathcal{F}) + 1}{1 - \gamma(\boldsymbol{\Delta})\gamma(\boldsymbol{\mathcal{M}})\,(\gamma(\mathcal{F}) + 1)}\right)||\mathbf{w}||\,. \tag{26}$$

Next, we plug the upper bound (26) into the inequality $||\mathbf{u}|| \leq \gamma(\boldsymbol{\mathcal{M}})||\widehat{\mathbf{w}}||$ to obtain

$$||\mathbf{u}|| \leq \left(\frac{\gamma(\boldsymbol{\mathcal{M}})\,(\gamma(\boldsymbol{\Delta})\gamma(\mathcal{F}) + 1)}{1 - \gamma(\boldsymbol{\Delta})\gamma(\boldsymbol{\mathcal{M}})(\gamma(\mathcal{F}) + 1)}\right)||\mathbf{w}||\,, \tag{27}$$

and subsequently, we plug (27) into the inequality $||\mathbf{x}|| \leq \gamma(\mathcal{F})(||\mathbf{u}|| + ||\mathbf{w}||)$ to obtain

$$||\mathbf{x}|| \leq \left(\gamma(\mathcal{F})\frac{1 + \gamma(\boldsymbol{\mathcal{M}})\,(1 - \gamma(\boldsymbol{\Delta}))}{1 - \gamma(\boldsymbol{\Delta})\gamma(\boldsymbol{\mathcal{M}})(\gamma(\mathcal{F}) + 1)}\right)||\mathbf{w}||\,. \tag{28}$$

The last step is to verify that the maps $\mathbf{w} \rightarrow \mathbf{x}$ and $\mathbf{w} \rightarrow \mathbf{u}$ have a finite $\mathcal{L}_p$-gain. This is done by checking that the gains in (27) and (28) are positive values when the gain of $\boldsymbol{\mathcal{M}}$ is sufficiently small. Since (22) holds, the denominator in (27) is positive. Since the numerator of (27) is always positive, we conclude that the map $\mathbf{w} \rightarrow \mathbf{u}$ has an $\mathcal{L}_p$-gain. Similarly for (28), since (22) implies that $\gamma(\boldsymbol{\mathcal{M}})\gamma(\boldsymbol{\Delta}) < 1$, we have that both numerator and denominator are positive. This implies that the map $\mathbf{w} \rightarrow \mathbf{x}$ has an $\mathcal{L}_p$-gain, as desired. ■

The robustness condition (22) highlights a trade-off between (*i*) the degree of tolerable uncertainty in the mismatch between nominal and real dynamics, and (*ii*) the extent of the set of stabilizing control policies that we are permitted to optimize over. Specifically, (22) ensures that, for any model mismatch $\boldsymbol{\Delta} \in \mathcal{L}_p$, there always exists a range of admissible gains for $\boldsymbol{\mathcal{M}}$ such that the closed-loop is stable. This enables one to freely learn over all appropriately gain-bounded operators. Further note that Theorem 3 is not conservative when $\boldsymbol{\Delta} = 0$—this is unlike the classical application of the small-gain theorem [42] which would enforce that $\gamma(\mathbf{K}) < (\gamma(\mathcal{F}))^{-1}$ even when $\boldsymbol{\Delta} = 0$. Indeed, when the model is fully known, the right-hand side of (22) diverges to infinity, allowing the gain of $\boldsymbol{\mathcal{M}}$ to be any finite value, although without imposing an upper bound, and therefore recovering the completeness result of Theorem 1. Last, we remark that the relationships (27) and (28) formally quantify the extent to which the model mismatch can deteriorate the amplification of disturbances on the closed loop trajectories $(\mathbf{x},\mathbf{u})$ for the system, for a given policy. However, it remains open how much the model uncertainty deteriorates the performance of the *optimal* policy. Such questions have only been rigorously answered for the linear-quadratic case, see, for instance, [43], [44].

Remark 1 (Robust stability of nonlinear SLS): The authors of [30] characterize robust stability of nonlinear SLS against mismatch in satisfying the achievability constraint (17b). Specifically, [30] focuses on the scenario where the control policy is a mapping $\mathbf{x} \rightarrow \mathbf{u}$ in the form

$$\tilde{\mathbf{w}} = \mathbf{x} - (\boldsymbol{\Psi}^{\mathbf{x}} - \mathbf{I})\tilde{\mathbf{w}}\,, \tag{29}$$

$$\mathbf{u} = \boldsymbol{\Psi}^{\mathbf{u}}(\tilde{\mathbf{w}})\,, \tag{30}$$

where $\tilde{\mathbf{w}}$ represent the internal state of the controller, for some $(\boldsymbol{\Psi}^{\mathbf{x}}, \boldsymbol{\Psi}^{\mathbf{u}}) \in \mathcal{L}_p$ which are not assumed to perfectly comply with (17b). Accordingly, the authors define a mismatch operator

$$\boldsymbol{\Xi} = \mathbf{F}(\boldsymbol{\Psi}^{\mathbf{x}}, \boldsymbol{\Psi}^{\mathbf{u}}) + \mathbf{I} - \boldsymbol{\Psi}^{\mathbf{x}}\,. \tag{31}$$

Then, Theorem IV.2 of [30] proves closed-loop stability as long as $\gamma(\boldsymbol{\Xi}) < 1$. Since $\boldsymbol{\Xi}$ measures the degree of violation of the achievability constraint rather than the degree of model

[6] For improving the clarity of the proof, from here onwards, we omit the subscript p of the signal norms.

uncertainty, a robust stability analysis based on verifying $\gamma(\boldsymbol{\Xi}) < 1$ tailored to the case $\mathbf{F} = \widehat{\mathbf{F}} + \boldsymbol{\Delta}$ may not be straightforward, and it is not attempted in [30]. For this case, instead, Theorem 3 provides an upper bound on the admissible gains for $\boldsymbol{\mathcal{M}}$; this is achieved by exploiting the IMC structure of the policy (23), and bounding the effect of model uncertainty on the closed-loop map for the ground-truth system.

B. DISTRIBUTED CONTROLLERS FOR LARGE-SCALE PLANTS

When dealing with large-scale cyber-physical systems, one may consider that the plant (1) is composed of a network of N dynamically interconnected nonlinear subsystems. To model this scenario, we introduce an undirected coupling graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V} = \{1, \ldots, N\}$ represent the subsystems in the network, and the set of edges $\mathcal{E}$ encode pairs of subsystems $\{i, j\}$ that are dynamically interconnected through state variables. Specifically, the dynamics of each subsystem $i \in \mathcal{V}$ is

$$x_t^{[i]} = f_t^{[i]}(x_{t-1:0}^{[\mathcal{N}_i]}, u_{t-1:0}^{[i]}) + w_t^{[i]}, \quad t = 1, 2, \ldots \tag{32}$$

where state and input of each subsystem $i \in \mathcal{V}$ at time $t = 1, 2, \ldots$ are denoted by $x_t^{[i]} \in \mathbb{R}^{n_i}$ and $u_t^{[i]} \in \mathbb{R}^{m_i}$ respectively, and the initial state is $x_0^{[i]} \in \mathbb{R}^{n_i}$. In operator form we have

$$\mathbf{x}^{[i]} = \mathbf{F}^{[i]}(\mathbf{x}^{[\mathcal{N}_i]}, \mathbf{u}^{[i]}) + \mathbf{w}^{[i]}, \tag{33}$$

where $\mathbf{F}^{[i]} : \ell^{n_{\mathcal{N}_i}} \times \ell^{m_i} \to \ell^{n_i}$. Note that, by stacking the subsystem dynamics in (32) together, we recover a system in the form (1), where $x_t = col_{i \in \mathcal{V}}(x_t^{[i]}) \in \mathbb{R}^n$, $u_t = col_{i \in \mathcal{V}}(u_t^{[i]}) \in \mathbb{R}^m$, and $w_t = col_{i \in \mathcal{V}}(w_t^{[i]}) \in \mathbb{R}^n$.

When controlling networked systems in the form (33), a common scenario is that the local feedback controller $u_t^{[i]}$ can only access information made available by its neighbors according to a communication network with the same topology of $\mathcal{G}$. This requirement translates into imposing the following additional constraint to the performance-boosting problem (Problem 1):

$$\mathbf{u}^{[i]} = \mathbf{K}^{[i]}(\mathbf{x}^{[\mathcal{N}_i]}), \quad \forall i \in \mathcal{V}. \tag{34}$$

The challenge becomes to parametrize only those stabilizing policies that are distributed according to (34). This can be achieved by exploiting the IMC controller architecture (11) in combination with the network sparsity of $\mathbf{F}$ highlighted in (33). Let us consider, for example, the networked plant of Fig. 3, where $\mathbf{u}^{[i]}$ depends on the local disturbance reconstructions $\widehat{\mathbf{w}}^{[i]}$ only, that is, $\mathbf{u}^{[i]} = \boldsymbol{\mathcal{M}}^{[i]}(\widehat{\mathbf{w}}^{[i]})$. In order to reconstruct $\widehat{\mathbf{w}}^{[1]}$, agent $i = 1$ needs to evaluate the local dynamics $\mathbf{F}^{[1]}(\mathbf{x}^{[1]}, \mathbf{x}^{[3]}, \mathbf{u}^{[1]})$; this, in turns, requires a measurement of the state $\mathbf{x}^{[3]}$ over time. Repeating this reasoning for the agents $i = 2$ and $i = 3$, one obtains an overall control policy $\mathbf{K}(\mathbf{x})$ whose agent-wise components are computed relying on measurements from neighboring subsystems only, thus complying with (34). We formalize this reasoning in the next proposition.

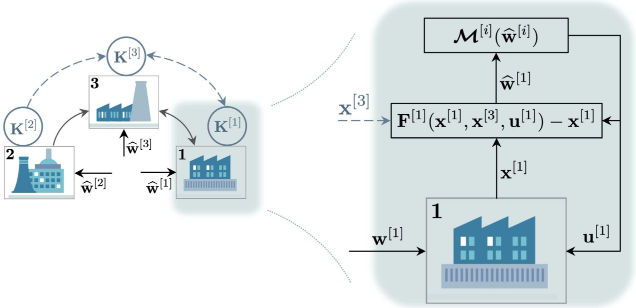
$\mathbf{K}^{[3]}$
$\mathbf{K}^{[2]}$
$\mathbf{K}^{[1]}$
3
2
1
$\widehat{\mathbf{w}}^{[3]}$
$\widehat{\mathbf{w}}^{[2]}$
$\widehat{\mathbf{w}}^{[1]}$
$\boldsymbol{\mathcal{M}}^{[i]}(\widehat{\mathbf{w}}^{[i]})$
$\widehat{\mathbf{w}}^{[1]}$
$\mathbf{x}^{[3]}$
$\mathbf{F}^{[1]}(\mathbf{x}^{[1]}, \mathbf{x}^{[3]}, \mathbf{u}^{[1]}) - \mathbf{x}^{[1]}$
$\mathbf{x}^{[1]}$
1
$\mathbf{w}^{[1]}$
$\mathbf{u}^{[1]}$

FIGURE 3. Example of networked dynamics (33) and decentralized IMC controller for agent $i = 1$.

Proposition 1: Let graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ describe the topology of a plant $\mathbf{F}$ as per (33). Consider an IMC control policy (11) where the operator $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ is *decentralized*, that is, $\boldsymbol{\mathcal{M}}^{[i]}(\widehat{\mathbf{w}}) = \boldsymbol{\mathcal{M}}^{[i]}(\widehat{\mathbf{w}}^{[i]})$ for every agent $i \in \mathcal{V}$. Then, the closed-loop system is ℓ_p-stable and the corresponding control policy $\mathbf{u} = \mathbf{K}(\mathbf{x})$ is distributed according to (34).

Proof: Since $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$, the closed-loop system is ℓ_p-stable by Theorem 1. By (33), we have $\widehat{\mathbf{w}}^{[i]} = \mathbf{x}^{[i]} - \mathbf{F}^{[i]}(\mathbf{x}^{[\mathcal{N}_i]}, \mathbf{u}^{[i]})$. Hence, agent i only needs measurements of the neighboring states according to $\mathcal{G}$ and local past inputs, thus complying with (34). ■

The result of Proposition 1 can be extended to more complex cases. First, one can use local operators $\boldsymbol{\mathcal{M}}^{[i]} \in \mathcal{L}_p$ that, besides $\widehat{\mathbf{w}}^{[i]}$, have access to disturbance reconstructions $\widehat{\mathbf{w}}^{[j]}$ or control variables $\mathbf{u}^{[j]}$ computed at locations $j \neq i$. While these architectures can be beneficial, e.g. for counteracting disturbances affecting other subsystems before they propagate to the subsystem i through coupling, they require additional communication channels $\{i, j\}$ if $j \notin \mathcal{N}_i$. Moreover, one has to use local operators $\boldsymbol{\mathcal{M}}^{[i]}$ guaranteeing that the whole operator $\boldsymbol{\mathcal{M}}$ belongs to $\mathcal{L}_p$. To this purpose, in general, it is not enough that $\boldsymbol{\mathcal{M}}^{[i]} \in \mathcal{L}_p$ because the dependency on $\widehat{\mathbf{w}}^{[j]}$ and $\mathbf{u}^{[j]}$ for $j \neq i$ can induce loop interconnections that can destabilize the closed-loop system. Classes of local operators $\boldsymbol{\mathcal{M}}^{[i]}$ yielding $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ have been proposed in [45], [46] by using dissipativity theory.

V. LEARNING TO BOOST PERFORMANCE USING UNCONSTRAINED OPTIMIZATION

Leveraging the theoretical results of previous sections, we reformulate the performance-boosting problem in a form that facilitates optimizing by automatic differentiation and unconstrained gradient descent. This enables the use of highly flexible cost functions for complex nonlinear optimal control tasks. By design, the proposed approach guarantees closed-loop stability throughout the optimization process. We assess the effectiveness of the proposed methodology in achieving optimal performance through numerical experiments, in Section VI.

A. IMC-BASED REFORMULATION OF PERFORMANCE BOOSTING

The main value of Theorem 1 is that it enables reformulating Problem 1 as follows.

IMC reformulation of the performance-boosting problem:

$$\min_{\mathcal{M}\in\mathcal{L}_p} \quad \mathbb{E}_{w_{T:0}}\left[L(x_{T:0}, u_{T:0})\right] \tag{35a}$$

$$\text{s. t.} \quad x_t = f_t(x_{t-1:0}, u_{t-1:0}) + w_t, \quad x_0 = w_0, \tag{35b}$$

$$u_t = \mathcal{M}_t(w_{t:0}), \quad t = 1, 2, \ldots . \tag{35c}$$

Indeed, (6) corresponds to (35b)–(35c). If the exact dynamics f_t in (35b) is not known, it must be simply replaced by the nominal model $\widehat{f}_t$.

The reformulation (35) offers significant computational advantages as compared to Problem 1. In the classical linear quadratic case,[7] (35) becomes strongly convex in $\mathcal{M}$—enabling to use efficient convex optimization for finding a globally optimal solution [37], [40], [47], [48], [49]. In the general nonlinear case, searching over nonlinear operators $\mathcal{M} \in \mathcal{L}_p$ remains significantly easier than tackling Problem 1 directly. Indeed, the set $\mathcal{K}$ of controllers $\mathbf{K}(\cdot)$ complying with (5b) is, in general, difficult to parametrize. This is mainly because, given two stabilizing policies $\mathbf{K}_1, \mathbf{K}_2$, their convex combinations $\mathbf{K}_3 = \gamma\mathbf{K}_1 + (1-\gamma)\mathbf{K}_2$ with $\gamma \in [0, 1]$ and their cascaded composition $\mathbf{K}_4 = \mathbf{K}_2(\mathbf{\Phi}^{\mathbf{x}}[\boldsymbol{F}, \mathbf{K}_1])$ do not result in stabilizing policies, in general; these issues are very well-known for the special case of linear systems [47], [50]. Hence, it is difficult to parameterize stabilizing policies, for instance, by composing or summing together base stabilizing operators. Instead, thanks to $\mathcal{L}_p$ being convex and closed under composition, there exist methods for parametrizing rich subsets of $\mathcal{L}_p$ through free parameters $\theta \in \mathbb{R}^d$, where $d \in \mathbb{N}$ is the number of scalar parameters, that is, to define operators $\mathcal{M}(\theta)$ such that

$$\mathcal{M}(\theta) \in \mathcal{L}_p, \quad \forall \theta \in \mathbb{R}^d . \tag{36}$$

This allows turning (35) into an unconstrained optimization problem over $\theta \in \mathbb{R}^d$.

The last issue to be addressed is the computation of the average in (35a) that, as noticed before, is generally intractable. This is usually circumvented by approximating the exact average with its empirical counterpart obtained using a set of samples $\{w^s_{T:0}\}_{s=1}^S$ drawn from the distribution $\mathcal{D}_{T:0}$. One then obtains the finite-dimensional optimization problem:

$$\min_{\theta\in\mathbb{R}^d} \quad \frac{1}{S}\sum_{s=1}^{S} L(x^s_{T:0}, u^s_{T:0}) \tag{37a}$$

$$\text{s. t.} \quad x^s_t = f_t(x^s_{t-1:0}, u^s_{t-1:0}) + w^s_t\,, \quad w^s_0 = x^s_0\,, \tag{37b}$$

$$u^s_t = \mathcal{M}_t(\theta)(w^s_{t:0})\,, \quad t = 0, 1, 2, \ldots , \tag{37c}$$

where $x^s_{T:0}$ and $u^s_{T:0}$ are the inputs and states obtained when the disturbance $w^s_{T:0}$ is applied. While in this work we only consider the empirical cost in the optimization problem (37a), the closed-loop performance when faced with out-of-sample noise sequences is further investigated in [51].

Finally, we highlight that (37b) and (37c) can be seen as the equations of the layer t of a neural network with T layers. Specifically, we can interpret the layer t of this neural network to have inputs $(x^s_{t-1:0}, u^s_{t-1:0}, w^s_{t:0})$ and outputs (x^s_t, u^s_t). Under this lens, the weights to be learned across all layers are the $\theta \in \mathbb{R}^d$ defining the control policy (37c). When $\mathcal{M}_t$, for $t = 0, 1, \ldots$ is sufficiently smooth, the absence of constraints on θ enables the use of powerful packages, such as TensorFlow [52] and PyTorch [53], leveraging automatic differentiation and backpropagation for optimizing the controller through gradient descent.

B. FREE PARAMETERIZATIONS OF $\mathcal{L}_2$ SUBSETS

As highlighted in Section V-A, the possibility of obtaining effective controllers by solving (37) critically depends on our ability to parametrize $\mathcal{L}_p$ operators. The main obstacle is that the space $\mathcal{L}_p$ is infinite-dimensional. Hence, for implementation, one usually restrict the search in subsets of $\mathcal{L}_p$ described by finitely many parameters. When linear systems are considered, one can search over Finite Impulse Response (FIR) transfer matrices $\mathbf{M} = \sum_{i=0}^N M[i]z^{-i} \in \mathcal{TF}_s$ and then optimize over the finitely many real matrices $M[i]$. Less and less conservative solutions can be obtained by increasing the FIR order N. However, the FIR approach limits the search to linear control policies.

Recently, [31], [32], [54] have proposed finite-dimensional DNN approximations of classes of nonlinear $\mathcal{L}_2$ operators. In the sequel we briefly review the Recurrent Equilibrium Network (REN) models proposed in [32]. An operator $\mathcal{M} : \ell^n \to \ell^m$ is a REN if the relationship $\mathbf{u} = \mathcal{M}(\widehat{\mathbf{w}})$ is recursively generated by the following dynamical system:

$$\begin{bmatrix} \xi_t \\ z_t \\ u_t \end{bmatrix} = \underbrace{\begin{bmatrix} A_1 & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}}_{W} \begin{bmatrix} \xi_{t-1} \\ \sigma(z_t) \\ w_t \end{bmatrix} + \underbrace{\begin{bmatrix} b_{x,t} \\ b_{z,t} \\ b_{w,t} \end{bmatrix}}_{b_t}, \quad \xi_{-1} = 0\,, \tag{38}$$

where $\xi_t \in \mathbb{R}^q$, $v_t \in \mathbb{R}^r$, $b_{x,t}, b_{z,t}, b_{w,t} \in \ell_\infty$[8] and $\sigma : \mathbb{R} \to \mathbb{R}$—the activation function—is applied element-wise. Further, $\sigma(\cdot)$ must be piecewise differentiable and with first derivatives restricted to the interval [0,1]. As noted in [32], RENs subsume many existing DNN architectures. In general, RENs define *deep equilibrium network* models [55] due to the implicit relationships defining z_t in the second block row of (38). By restricting D_{11} to be strictly lower-triangular, the value of z_t can be computed explicitly, thus significantly speeding-up computations [32]. To give an example of the expressivity of (38), by suitably choosing the size and zero pattern of matrices

[7] That is, when f_t and $\mathcal{M}$ are linear and L is quadratic positive definite.

[8] This is slightly different from the original REN model, where these signals [32] are assumed to be constant.

in (38), RENs can provide nonlinear systems in the form

$$\begin{aligned} \xi_t &= \hat{A}\xi_{t-1} + \hat{B}\,\mathrm{NN}^{\xi}(\xi_{t-1}, \widehat{w}_t) \\ u_t &= \hat{C}\xi_t + \hat{D}\,\mathrm{NN}^{u}(\xi_{t-1}, \widehat{w}_t) \end{aligned}$$

where $\hat{A}, \hat{B}, \hat{C}, \hat{D}$ are arbitrary matrices of suitable dimensions and $NN^{\star}$, $\star \in \{\xi, u\}$, are neural networks of depth L given by the relations

$$\begin{aligned} \tilde{z}^{\star}_{0,t} &= [\xi_{t-1}^{\top}, \hat{w}_t^{\top}]^{\top}, \\ \tilde{z}^{\star}_{k+1,t} &= \sigma(W^{\star}_k \tilde{z}^{\star}_{k,t} + b^{\star}_k), \quad k = 0, \ldots L-1 \end{aligned}$$

where $W^{\star}_k$ and $b^{\star}_k$ are the layer weights and biases, respectively, and $\tilde{z}^{\star}_{L,t}$ is the NN output.

For an arbitrary choice of W and b_t, the map $\mathcal{M}$ induced by (38) may not lie in $\mathcal{L}_2$. The work [32] provides an explicit smooth mapping $\Theta : \mathbb{R}^d \to \mathbb{R}^{(q+r+m)\times(q+r+n)}$ from unconstrained training parameters $\theta \in \mathbb{R}^d$ to a matrix $W = \Theta(\theta) \in \mathbb{R}^{(q+r+m)\times(q+r+n)}$ defining (38), with the property that the corresponding operator $\mathcal{M}(\theta)$ lies in $\mathcal{L}_2$ by design when $b_t = 0$.[9] This approach can be easily generalized by including vectors $b_t, t = 1, \ldots, T$ in the set of trainable parameters and assuming $b_t = 0$ for $t > T$.

Recently, free parameterizations of continuous-time $\mathcal{L}_2$ operators through RENs and port-Hamiltonian systems have been also proposed in [54] and [56], respectively.

Remark 2: The work [19] proves that RENs in the form (38) are universal approximators of all contracting and Lipschitz operators when the parameters (W, b) do not vary with time. To the best of the authors' knowledge, it is still unknown if the class of RENs in $\mathcal{L}_2$, parametrized by $W = \Theta(\theta)$ where $W \in \mathbb{R}^{(q+r+m)\times(q+r+n)}$, can approximate any operator in $\mathcal{L}_2$ arbitrarily well. Our work motivates future research efforts to discover new parametrizations of operators in $\mathcal{L}_p$ with stronger and provable approximation capabilities.

To conclude, we clarify that RENs can be directly embedded into the performance-boosting optimization problem (37a)–(37c). This is obtained by substituting the input (37c) with the recursions (38), where $W = \Theta(\theta)$ according to the mapping proposed in [32].

VI. NUMERICAL EXPERIMENTS: THE MAGIC OF THE COST

In this section, we test the flexibility of performance boosting by considering cooperative robotics problems. Firstly, we validate the guarantees of the design approach by showing that closed-loop stability is preserved during and after training—both when the system model is known and when it is uncertain. Secondly, we exploit the freedom in selecting the cost $L(x_{T:0}, u_{T:0})$ to include appropriate terms aimed at promoting complex closed-loop behaviors.

In all the examples, we consider two point-mass vehicles, each with position $p^{[i]}_t \in \mathbb{R}^2$ and velocity $q^{[i]}_t \in \mathbb{R}^2$, for $i = 1, 2$, subject to nonlinear drag forces (e.g., air or water resistance). The discrete-time model for vehicle i is

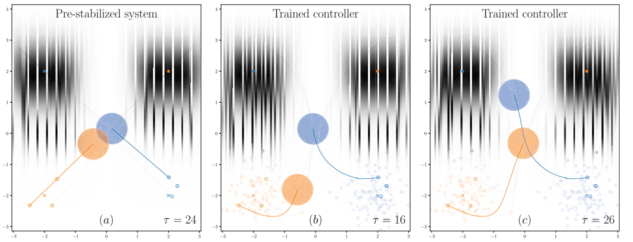

FIGURE 4. Mountains—Closed-loop trajectories before training (left) and after training (middle and right) over 100 randomly sampled initial conditions marked with ∘. Snapshots taken at time-instants τ. Colored (gray) lines show the trajectories in $[0, \tau_i]$ ($[\tau_i, \infty)$). Colored balls (and their radius) represent the agents (and their size for collision avoidance).

$$\begin{bmatrix} p^{[i]}_t \\ q^{[i]}_t \end{bmatrix} = \begin{bmatrix} p^{[i]}_{t-1} \\ q^{[i]}_{t-1} \end{bmatrix} + T_s \begin{bmatrix} q^{[i]}_{t-1} \\ (m^{[i]})^{-1}\left(-C(q^{[i]}_{t-1}) + F^{[i]}_{t-1}\right) \end{bmatrix}, \tag{39}$$

where $m^{[i]} > 0$ is the mass, $F^{[i]}_t \in \mathbb{R}^2$ denotes the force control input, $T_s > 0$ is the sampling time and $C^{[i]} : \mathbb{R}^2 \to \mathbb{R}^2$ is a *drag function* given by $C^{[i]}(s) = b^{[i]}_1 s - b^{[i]}_2 \tanh(s)$, for some $0 < b^{[i]}_2 < b^{[i]}_1$. Each vehicle must reach a target position $\overline{p}^{[i]} \in \mathbb{R}^2$ with zero velocity in a stable way. This elementary goal can be achieved by using a base proportional controller

$$F'^{[i]}_t = K'^{[i]}(\bar{p}^{[i]} - p^{[i]}_t), \tag{40}$$

with $K'^{[i]} = \mathrm{diag}(k^{[i]}_1, k^{[i]}_2)$ and $k^{[i]}_1, k^{[i]}_2 > 0$. The overall dynamics $f_t(x_{t-1:0}, u_{t-1:0})$ in (1) is given by (39)–(40) with

$$F^{[i]}_t = F'^{[i]}_t + u^{[i]}_t, \tag{41}$$

where $x_t = (p^{[1]}_t, q^{[1]}_t, p^{[2]}_t, q^{[2]}_t)$ and $u_t = (u^{[1]}_t, u^{[2]}_t)$ is a performance-boosting control input to be designed. As per (1), we consider additive disturbances affecting the system dynamics. Thanks to the use of the prestabilizing controller (40), one can show that $\mathcal{F}(\mathbf{u}, \mathbf{w}) \in \mathcal{L}_2$.

The goal of the performance-boosting policy is to enforce additional desired behaviors, on top of stability, which are specified in each of the following subsections. In all cases, we parametrize the operator $\mathcal{M}(\theta) \in \mathcal{L}_2$ as a REN, see (38). Appendix A presents all the implementation details, such as parameter values and exact definitions of the cost functions. Appendix B compares the performance of our methods and the corresponding guarantees with two related baseline approaches. The code to reproduce our examples as well as various movies are available in our Github repository.[10]

A. ROBUST STABILITY PRESERVATION DURING OPTIMIZATION

We consider the scenario `mountains` in Fig. 4 where each vehicle must reach the target position in a stable way while

[9] Furthermore, RENs enjoy contractivity – although the theoretical results of this paper do not rely on this property.

[10] https://github.com/DecodEPFL/performance-boosting_controllers.git

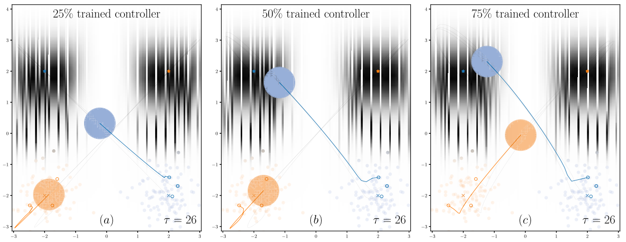

FIGURE 5. Mountains—Closed-loop trajectories after 25%, 50% and 75% of the total training whose closed-loop trajectory is shown in Fig. 4. Even if the performance can be further optimized, stability is always guaranteed.

avoiding collisions between themselves and with two grey obstacles. Each agent is represented with a circle that indicates its radius for the collision avoidance specifications. When using the base controller (40), the vehicles successfully achieve the target, however, they do so with poor performance since collisions are not avoided, as shown in Fig. 4(a).

We select a loss $L(x_{T:0}, u_{T:0})$ as the sum of stage costs $l(x_t, u_t)$, that is, $L(x_{T:0}, u_{T:0}) = \sum_{t=0}^{T} l(x_t, u_t)$ with

$$l(x_t, u_t) = l_{traj}(x_t, u_t) + l_{ca}(x_t) + l_{obs}(x_t)\,, \tag{42}$$

where $l_{traj}(x_t, u_t) = \begin{bmatrix} x_t^\mathsf{T} & u_t^\mathsf{T} \end{bmatrix} Q \begin{bmatrix} x_t^\mathsf{T} & u_t^\mathsf{T} \end{bmatrix}^\mathsf{T}$ with $Q \succeq 0$ penalizes the distance of agents from their targets and the control energy, $l_{ca}(x_t)$ and $l_{obs}(x_t)$ penalize collisions between agents and with obstacles, respectively.

In order to train the performance-boosting controller, we solve (37), using a REN (38) of dimension $q = r = 8$. The training data consists of a set of 100 initial positions, i.e., we set $w_0 = ((p_0^x)^{[1]}, (p_0^y)^{[1]}, 0, 0, (p_0^x)^{[2]}, (p_0^y)^{[2]}, 0, 0)$ and $w_t = 0$, for $t > 0$, where p^x and p^y denote the x and y coordinates of the vehicles in the Cartesian plane, respectively. Initial positions are sampled from a Gaussian distribution around the nominal initial condition. Fig. 4(b) and (c) shows the nominal and training initial conditions marked with '×' and '∘', respectively, and three test trajectories after the training of the IMC controller. The trained control policies avoid collisions and achieve optimized trajectories thanks to minimizing (42).

1) EARLY STOPPING OF THE TRAINING

We validate the stability-by-design property of our IMC control policies. We consider the scenario `mountains` as above but where the training process is interrupted before achieving a local minimum, as per the one in Fig. 4. In particular, we stop the optimization algorithm after 25%, 50%, and 75% of the total number of epochs. The obtained trajectories are shown in Fig. 5. We observe that even if the performance is not optimized, closed-loop stability is always guaranteed.

2) MODEL MISMATCH

We test our trained IMC controller when considering model mismatch on the system. In particular, we assume that the true vehicles have an incertitude over the mass of ±10%, and we apply IMC control policies embedding the nominal system with the nominal mass value. Fig. 6(a) and (b) validate the robust ℓ_2-stability of the closed-loop trajectories when the vehicles are lighter and heavier, respectively. Theorem 3 suggests that, in this case, the gain of $\boldsymbol{\mathcal{M}}$ may be sufficiently low to counteract the effect of model uncertainty. Note, however, that checking the sufficient condition (22) requires computing an upper bound on $\gamma(\boldsymbol{\Delta})$—a cumbersome task for general nonlinear systems. Nonetheless, Theorem 3 ensures that, in practical implementation, we can always reduce $\gamma(\boldsymbol{\mathcal{M}})$ enough to eventually meet (22).

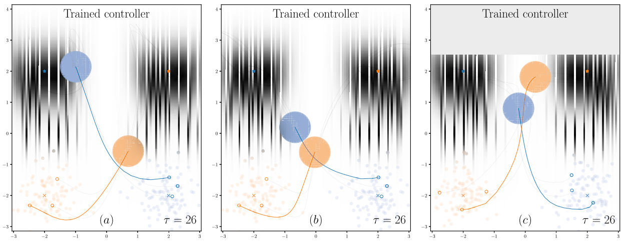

FIGURE 6. Mountains—Closed-loop trajectories after training. (Left and middle) Controller tested over a system with mass uncertainty (-10% and +10%, respectively). (Right) Trained controller with safety promotion through (45). Training initial conditions marked with ∘. Snapshots taken at time-instants τ. Colored (gray) lines show the trajectories in $[0, \tau_i]$ ($[\tau_i, \infty)$). Colored balls (and their radius) represent the agents (and their size for collision avoidance).

B. BOOSTING FOR SAFETY AND INVARIANCE CERTIFICATES

A challenging task in many control applications is to deal with stringent safety constraints on the state variables. Ideally, one would directly add the constraint that

$$x_t \in \mathcal{C}\,, \forall t = 0, 1, \ldots\,, \tag{43}$$

in the IMC-based performance-boosting problem (35), where $\mathcal{C} \subseteq \mathbb{R}^n$ defines a safety region. Unfortunately, (43) generally results in intractable constraints over $\boldsymbol{\mathcal{M}}$. Indeed, it may be challenging to even verify that (43) holds for a certain $\boldsymbol{\mathcal{M}}$ due to the infinite-horizon requirement and the involved nonlinearities. Many state-of-the-art approaches for guaranteeing safety hinge on either predictive safety filters [57], [58] or Control Barrier Functions (CBFs) [59], [60]. Safety filters are used during deployment: they override the control input $\mathbf{u} = \boldsymbol{\mathcal{M}}(\widehat{\mathbf{w}})$ with a different (suboptimal) control variable when deemed necessary for guaranteeing safety. Instead, CBFs can be used for safety verification of a given policy, as they allow characterizing $\mathcal{C}$ as a forward invariant set based on a safety-set-defining function $h(x) : \mathcal{X} \rightarrow \mathbb{R}$ satisfying $h(x) \geq 0$ for all $x \in \mathcal{C}$. Certifying the forward invariance of $\mathcal{C}$ translates into determining if $h(x)$ is a CBF through verification of some safety conditions.[11] In particular, one can verify that, for any

[11] An exact definition of CBFs for the discrete-time can be found in [60]; for a more general discussion on CBFs we refer the reader to [59].

$x_t \in \mathcal{C}$, if there exists an input u_t giving x_{t+1} such that it holds

$$h(x_{t+1}) - h(x_t) + \gamma h(x_t) \geq 0\,, \tag{44}$$

where $0 < \gamma \leq 1$, then $h(x)$ is a CBF.

While optimizing over $\boldsymbol{\mathcal{M}}$ such that (43) holds by design remains an open challenge, we aim to promote forward invariant sets by shaping the cost to include soft safety specifications over a horizon of length T. In particular, the new cost term penalizes violations of (44) as per

$$L_{\text{inv}} = \sum_{t=0}^{T-1} \text{ReLU}\left(h(x_t) - h(x_{t+1}) + \gamma h(x_t)\right)\,. \tag{45}$$

We consider the `mountains` scenario again and add the requirement that $(p_t^y)^{[i]} < (\bar{p}^y)^{[i]} + 0.1$ for each vehicle $i = 1, 2$ and every $t = 0, 1, \ldots$, where p_t^y denotes the y-coordinate of each center-of-mass position on the Cartesian plane. In other words, we only allow an overshoot of 0.1in the vertical direction with respect to the target position for each vehicle. By defining $h(x_t) = \sum_{i=1}^{2}((\bar{p}^y)^{[i]} + 0.1 - (p_t^y)^{[i]})$ we add the term (45) to the loss function (37a). Upon training without including L_{inv} in the cost, the masses violate the constraints, on average, on 67.49% of the time over 100 runs—typical trajectories are shown in Fig. 4. The violation ratio is decreased to 5.43% when L_{inv} is included, as shown in Fig. 6(c), where the gray area indicates the unsafe region to be avoided by the vehicles. Note that shaping the cost through L_{inv} is also beneficial if one implements an online safety filter such as [57], [58] during deployment. This is because penalizing L_{inv} drastically decreases constraint violations of the closed-loop system, and hence, the suboptimal online intervention of the safety filter would be much less frequent.

C. BOOSTING FOR TEMPORAL LOGIC SPECIFICATIONS

The success of many policy learning algorithms, e.g., in RL, is highly dependent on the choice of the reward functions for capturing the desired behavior and constraints of an agent. When tasks become complex, specifying loss functions that are the sum over time of stage costs can be restrictive. For instance, consider the case of an agent that must optimally visit a set of locations. A loss function composed of a stage-cost summed over time—that is, the one considered in dynamic programming and classical optimal control [3], [61]—cannot easily capture this task, as it would need a-priori information about the optimal timings to visit each location. To overcome this problem, one could use more complex loss functions, as per those derived from temporal logic formulations. In particular, truncated linear temporal logic (TLTL) is a specification language leveraging a set of operators defined over finite-time trajectories [62], [63]. It allows incorporating domain knowledge, and constraints (in a soft fashion) into the learning process, such as "always avoid obstacles", "eventually visit location a", or "do not visit location b until visiting location a". Then, using quantitative semantics one can automatically transform TLTL formulae into real-valued loss functions that are compositions of min and max functions over a finite period of time [62], [63].

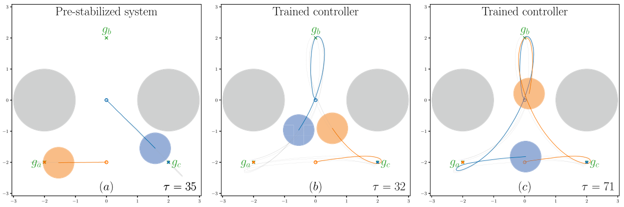

FIGURE 7. Waypoint-tracking—Closed-loop trajectories before training (left) and after training (middle and right). Snapshots taken at time-instants τ. Colored (gray) lines show the trajectories in $[0, \tau_i]$ ($[\tau_i, \infty)$). Colored balls (and their radius) represent the agents (and their size for collision avoidance).

To test the efficacy of TLTL specifications for shaping complex stable closed-loop behavior, we consider the scenario `waypoint-tracking`, shown in Fig. 7, where the two vehicles have to visit a sequence of waypoints while avoiding collisions between them and the gray obstacles. The *blue* vehicle's goal is to visit g_b, then g_a and then g_c, while the goal for the *orange* vehicle is to visit the waypoints in the following order: g_c, g_b and g_a. Following [62], the loss formulation for the *orange* agent is translated into plain English as "*Visit g_c then g_b then g_a; and don't visit g_b or g_a until visiting g_c; and don't visit g_a until visiting g_b; and if visited g_c, don't visit g_c again; and if visited g_b, don't visit g_b again; and always avoid obstacles; and always avoid collisions; and eventually state at the final goal.* " Its mathematical formulation can be found in Appendix A-2.

Fig. 7 shows the `waypoint-tracking` scenario before and after the training of a performance-boosting controller. As described in Section V-B, we use a REN with $q = r = 32$ for approximating the $\mathcal{L}_2$ operator $\boldsymbol{\mathcal{M}}$. Furthermore, we allow for a time-varying bias of the form $b_t^\top = \begin{bmatrix} 0_{1\times q} & 0_{1\times r} & b_{w,t}^\top \end{bmatrix}$, in (38), with $b_{w,t} = 0$ for $t > T$. While the system always starts at the same initial condition indicated with '$\circ$', the data consists of disturbance sequences $w_{T:0}$ with fixed w_0 and $w_{T:1}$ as i.i.d. samples drawn from a Gaussian distribution with zero mean and standard deviation of 0.01. Our result highlights the power of complex costs—expressed through the TLTL loss function—which promotes vehicles visiting the predefined waypoints in the correct order while avoiding collisions between them and with the obstacles.

VII. CONCLUSION

Embedding safety and stability emerges as a crucial challenge when control systems are equipped with high-performance machine learning components. This work aims to contribute to this rapidly developing field by uncovering the theoretical and computational potential of IMC for safely boosting the performance of closed-loop nonlinear systems with machine learning models such as DNNs.

The results of this work open up several future research directions. First, motivated by the recent results of [51], it would be relevant to apply statistical learning theory to rigorously assess the generalization capabilities of performance-boosting controllers in uncertain environments, with uncertain models, and over extended time frames. Second, drawing on insights from [64], integrating extensive RL-based offline learning with real-time adjustments similar to MPC presents a promising approach. Third, within the IMC framework, there is a significant opportunity to develop richer parametrizations of stable dynamical systems in $\mathcal{L}_p$, and to theoretically prove their approximation capabilities. Lastly, building upon [65], it is interesting to explore how learning-based IMC methods could generate new optimization algorithms with formal guarantees for tackling complex optimal control and machine learning tasks.

APPENDIX

A. IMPLEMENTATION DETAILS FOR THE NUMERICAL EXPERIMENTS IN SECTION VI

We set $m^{[i]} = b_1^{[i]} = k'^{[i]}_1 = k'^{[i]}_2 = 1$ and $b_2^i = 0.5$ as the parameters for each vehicle i, in the model (39) with the pre-stabilizing controller (40). The collision-avoidance radius of each agent is 0.5.

1) MOUNTAINS SCENARIO

As shown in Fig. 4, the vehicles start at $p_0^{[1]} = (-2, -2)$ and $p_0^{[2]} = (-2, 2)$, and their goal is to go to the target positions $\bar{p}^{[1]} = (2, 2)$ and $\bar{p}^{[2]} = (-2, 2)$, respectively. The training data consists of 100 initial positions sampled from a Gaussian distribution around the initial position with a standard deviation of 0.5.

Let $\bar{x} = (\bar{x}^{[1]}, \bar{x}^{[2]})$ with $\bar{x}^{[i]} = (\bar{p}^{[i]}, 0_2)$. The terms of the cost function (42) are defined as follows:

$$l_{traj}(x_t, u_t) = (x_t - \bar{x})^\top \tilde{Q}(x_t - \bar{x}) + \alpha_u u_t^\top u_t$$

$$l_{ca}(x_t) = \begin{cases} \alpha_{ca}\sum_{i=0}^{N}\sum_{j,\, i\neq j}(d_t^{i,j}+\epsilon)^{-2} & \text{if } d_t^{i,j} \leq D_{\text{safe}}\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

where $\tilde{Q} \succ 0$ and $\alpha_u, \alpha_{ca} > 0$ are hyperparameters, $d_t^{i,j} = |p_t^{[i]} - p_t^{[j]}|_2 \geq 0$ denotes the distance between agent i and j, $\epsilon > 0$ is a fixed positive small constant such that the loss remains bounded for all distance values and D_{safe} is a safe distance between the center of mass of each the agent; we set it to 1.2.

Motivated by [66], we represent the obstacles based on a Gaussian density function

$$\eta(z;\, \mu,\, \Sigma) = \frac{1}{2\pi\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(z-\mu)^\top \Sigma^{-1}(z-\mu)\right),$$

with mean $\mu \in \mathbb{R}^2$ and covariance $\Sigma \in \mathbb{R}^{2\times 2}$ with $\Sigma \succ 0$. The term $l_{obs}(x_t)$ is given by

$$\begin{aligned} l_{obs}(x_t) = \alpha_{obs}\sum_{i=0}^{2}\Bigg(&\eta\left(p_t^{[i]}; \begin{bmatrix} 2.5 \\ 0 \end{bmatrix}, 0.2\, I\right) \\ &+ \eta\left(p_t^{[i]}; \begin{bmatrix} -2.5 \\ 0 \end{bmatrix}, 0.2\, I\right) \\ &+ \eta\left(p_t^{[i]}; \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}, 0.2\, I\right) \\ &+ \eta\left(p_t^{[i]}; \begin{bmatrix} -1.5 \\ 0 \end{bmatrix}, 0.2\, I\right)\Bigg). \qquad (46) \end{aligned}$$

TABLE 1. Predicates Used in the TLTL Formulation of (47)

Predicates	Expression
ψ_{g_1}	$d^{g_1} < 0.05$
ψ_{g_2}	$d^{g_2} < 0.05$
ψ_{g_3}	$d^{g_3} < 0.05$
ψ_{o_1}	$d^{o_1} > r_{obs}$
ψ_{o_2}	$d^{o_2} > r_{obs}$
ψ_{coll}	$d^{rob} > 2\, r_{rob}$

For the hyperparameters, we set $\alpha_u = 2.5 \times 10^{-4}$, $\alpha_{ca} = 100$, $\alpha_{obs} = 5 \times 10^3$ and $Q = I_4$. We use stochastic gradient descent with Adam to minimize the loss function, setting a learning rate of 1×10^{-4}. We train for 5×10^3 epochs with one trajectory per batch size.

2) WAYPOINT-TRACKING SCENARIO

As shown in Fig. 4, the vehicles start at $p_0^{[1]} = (-2, 0)$ and $p_0^{[2]} = (0, 0)$. The goal points g_a, g_b and g_c are located at $(-2, -2)$, (0,2) and $(2, -2)$, respectively. To describe the TLTL loss, let us define, for each vehicle, the following functions of time:

- $d_t^{g_i}$, for $i = 1, 2, 3$, is the distance between the vehicle and the goal point g_i;
- $d_t^{o_i}$, for $i = 1, 2$, is the distance between the vehicle and the i^{th} obstacle;
- d_t^{coll} is the distance between the two vehicles;

where g_1, g_2 and g_3 are the waypoints in the correct visiting order, for each vehicle. Following the notation of [62], the temporal logic form of the cost function, for each vehicle, is

$$\begin{aligned} &(\psi_{g_1}\,\mathcal{T}\psi_{g_2}\,\mathcal{T}\psi_{g_3}) \wedge (\neg(\psi_{g_2} \vee \psi_{g_3})\,\mathcal{U}\,\psi_{g_1}) \wedge (\neg\psi_{g_3}\,\mathcal{U}\,\psi_{g_2}) \\ &\wedge \left(\bigwedge_{i=1,2,3} \square\,(\psi_{g_i} \Rightarrow \bigcirc\square\neg\psi_{g_i})\right) \wedge \left(\bigwedge_{i=1,2} \square\psi_{o_i}\right) \\ &\wedge \square\psi_{coll} \wedge \Diamond\square\psi_{g_3} \qquad (47) \end{aligned}$$

where ψ are predicates defined in Table 1 , and $r_{obs} = 1.7$ and $r_r = 0.5$ are the radii of the obstacles and vehicles, respectively.[12] The Boolean operators ¬, ∨, and ∧ stand for negation (not), disjunction (or), and conjunction (and). The temporal operators $\mathcal{T}$, $\mathcal{U}$, $\Diamond$, and $\square$ stand for 'then', 'until', 'eventually', and 'always'. Mathematically, each term can be automatically

[12] Note that in the waypoint-tracking scenario, we do not model the obstacles with a Gaussian density function.

translated following [62], [63]. For instance, $\square\psi_{coll}$ translates into

$$\min_{t\in[0,T]} (d_t^{rob} - 2r_{rob}),$$

and $\square(\psi_{g_i} \Rightarrow \bigcirc\square\neg\psi_{g_i})$ translates into

$$\min_{t\in[0,T]} \max\left(-(0.05 - d_t^{g_i}),\quad \min_{\tilde{t}\in[t+1,T]} -(0.05 - d_t^{g_i})\right).$$

The full mathematical expression of (47), which can be obtained following [62], is implemented in our Github repository.

We also add a small regularization term for promoting that the vehicles stay close to the end target point, which reads $\alpha_{\text{reg}}\|x_t - \bar{x}\|^2$, with $\alpha_{\text{reg}} = 1 \times 10^{-4}$. We use stochastic gradient descent with Adam to minimize the loss function, setting a learning rate of 5×10^{-4}. We train for 3000 epochs with a single trajectory per batch size.

B. COMPARISON OF PERFORMANCE-BOOSTING CONTROLLERS WITH OTHER BASELINES

We compare the performance of our proposed controllers with two baseline approaches for the scenario mountains presented in Section VI-A. In both cases, the vehicles are equipped with the base proportional controller (40) which is able to steer the agents towards the target position in a stable way. As described in Section VI-A, improving the performance means vehicles must avoid collisions with each other and with obstacles.

The first baseline we consider is a control policy derived by solving an optimization problem in a receding-horizon manner. This optimization problem is defined over the set of control inputs that ensure collision avoidance within the horizon.

The second baseline is to directly parametrize the entire control policy $\mathbf{u} = \mathbf{K}(x)$ as a recurrent neural network, that is, without adopting the IMC architecture of Fig. 1 train a control policy $\mathbf{u} = \mathbf{K}(\mathbf{x})$ directly parametrized as a recurrent neural network optimizing the cost $L(x_{T:0}, u_{T:0})$ defined in Section VI-A. Note that this approach does not guarantee the stability of the resulting closed-loop system.

1) ONLINE-OPTIMIZATION USING BARRIER FUNCTIONS OVER THE BASE CONTROLLER

A common approach in robotics for avoiding collisions and unsafe regions is to use control barrier functions [59], [60].

This requires online optimization for computing the system inputs. Specifically, we consider the approach in [60] for guaranteeing that the safe region is forward invariant. The online optimization problem reads as

$$u_t^* = \arg\min_{u_t, u_{t+1}} u_t^\top u_t \tag{48a}$$

$$\text{s.t.} x_t = (p_t^{[1]}, q_t^{[1]}, p_t^{[2]}, q_t^{[2]}), \tag{48b}$$

$$u_t = (u_t^{[1]}, u_t^{[2]}),\ (39), (40), (41), \tag{48c}$$

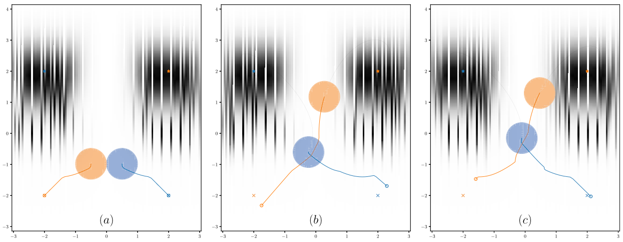

FIGURE 8. Mountains—Closed-loop trajectories when using the online policy given by (48). Snapshots of three trajectories starting at different test initial conditions.

$$h(x_{t+1}) - h(x_t) + \gamma\, h(x_t) \geq 0, \tag{48d}$$

$$h(x_{t+2}) - h(x_{t+1}) + \gamma\, h(x_{t+1}) \geq 0, \tag{48e}$$

where $0 < \gamma \leq 1$ and u_t^* is the safety-preserving input to the system. The barrier function $h : \mathbb{R}^n \to \mathbb{R}$ characterizes the region $\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}$ in the state space where no collisions between agents nor with the obstacles occur. To this purpose, we define

$$h(x) = \left(|p^{[1]} - p^{[2]}|^2 - 4\, r_{\text{agent}}^2\right) + \sum_{i=1}^{2}\sum_{j=1}^{4}\left(|p^{[i]} - p^{\text{obs}_j}|^2 - r_{\text{obs}}^2\right),$$

which is positive in the safe region. The radius of each agent is $r_{\text{agent}} = 0.5$, while $r_{\text{obs}} = 1.4$ denotes the radius of two obstacles, modeled as disks. The center of each disk is given by $p^{\text{obs}_j} \in \mathbb{R}^2$ and is the mean of the Gaussian density functions used for defining l_{obs} in (46). In the absence of collisions at time t, the constraint (48e) forces the agents to stay in the safe region at time $t + 1$ as well.

Fig. 8 shows three closed-loop trajectories of the agents when starting from different initial conditions. When the initial positions are symmetric with respect to the y-axis (Fig. 8(a)), the optimization problem (48) cannot find an input u_t^* allowing both agents to pass through the narrow corridor, and the agents stop without reaching the target. This is due to the reactive nature of CBFs, which do not account for the behavior of the system nor prioritize target reaching. To provide an objective performance assessment, we compare the quadratic cost term on the state, i.e. we evaluate the Euclidean distance to the target using

$$\tilde{L}(x_{T:0}, u_{T:0}) = \sum_{t=0}^{T}(x_t - \bar{x})^\top(x_t - \bar{x}), \tag{49}$$

over 20 test initial conditions (sample trajectories are displayed in Fig. 8(b) and (c)). The average cost incurred by the control law is 25.81, while it is 20.94 when using our approach. We highlight that when the vehicles are close enough to their respective target positions, one has $u_t^\star = 0$, and the system inherits the stability properties due to the base controller.

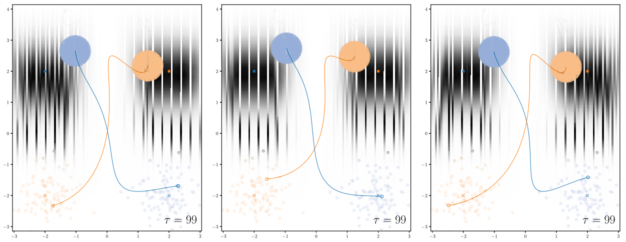

FIGURE 9. **Mountains—Three different closed-loop trajectories after training a REN controller without $\mathcal{L}_2$ stability guarantees over 100 randomly sampled initial conditions marked with ∘. Colored (gray) lines show the trajectories in (after) the training time interval.**

2) A RECURRENT NEURAL NETWORK CONTROLLER

We replace the controller in Fig. 1 by a REN where the trainable parameters are the weights W and the time-invariant bias $b_t = b$ in (38). Note that we do not constrain the REN to be an $\mathcal{L}_2$ operator, i.e., we do not use the mapping Θ described in Section V-B for redefining the trainable parameters. The model consists of 861 parameters which are optimized for minimizing the cost $L(x_{T:0}, u_{T:0})$, using the same initial conditions as in the experiments of Section VI-A. Fig. 9 shows three closed-loop trajectories of the agents when starting from different initial positions. Note that the targets are no longer the equilibria of the closed-loop system, and the vehicles move away from the targets after an initial reaching phase. The cost (49) incurred by this control law is 26.60, while it is 20.94 when using a performance-boosting controller (where the REN representing the operator $\boldsymbol{\mathcal{M}}$ has 864 parameters, i.e., only three more than the above REN controller).

REFERENCES

[1] A. M. Annaswamy, K. H. Johansson, and G. J. Pappas, “Control for societal-scale challenges: Road map 2030,” *IEEE Control Syst. Mag.*, vol. 44, no. 3, pp. 30–32, Jun. 2024.

[2] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*, vol. 10. Berlin, Germany: Springer Science & Business Media, 2013.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I-II. Belmont, MA, USA: Athena Scientific, 2011.

[4] L. S. Pontryagin, *Mathematical Theory of Optimal Processes*. Evanston, IL, USA: Routledge, 2018.

[5] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. San Francisco, CA, USA: Nob Hill Publishing, 2017.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[7] L. Brunke et al., “Safe learning in robotics: From learning-based control to safe reinforcement learning,” *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, pp. 411–444, 2022.

[8] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.

[9] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, “Autonomous drone racing with deep reinforcement learning,” in *2021 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1205–1212.

[10] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[11] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, “Safe model-based reinforcement learning with stability guarantees,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 909–919.

[12] M. Zanon and S. Gros, “Safe reinforcement learning using robust MPC,” *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[13] M. Jin and J. Lavaei, “Stability-certified reinforcement learning: A control-theoretic perspective,” *IEEE Access*, vol. 8, pp. 229086–229100, 2020.

[14] T. Parisini and R. Zoppoli, “A receding-horizon regulator for nonlinear systems and a neural approximation,” *Automatica*, vol. 31, no. 10, pp. 1443–1451, Oct. 1995.

[15] T. Parisini, M. Sanguineti, and R. Zoppoli, “Nonlinear stabilization by receding-horizon neural regulators,” *Int. J. Control*, vol. 70, no. 3, pp. 341–362, Jan. 1998.

[16] A. Levin and K. Narendra, “Control of nonlinear dynamical systems using neural networks. II. Observability, identification, and control,” *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 30–42, Jan. 1996.

[17] F. Gu, H. Yin, L. El Ghaoui, M. Arcak, P. Seiler, and M. Jin, “Recurrent neural network controllers synthesis with stability guarantees for partially observed systems,” in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 5385–5394.

[18] P. Pauli, J. Köhler, J. Berberich, A. Koch, and F. Allgöwer, “Offset-free setpoint tracking using neural network controllers,” in *Proc. Int. Conf. Learn. Dyn. Control*, 2021, pp. 992–1003.

[19] R. Wang, N. H. Barbara, M. Revay, and I. Manchester, “Learning over all stabilizing nonlinear controllers for a partially-observed linear system,” *IEEE Control Syst. Lett.*, vol. 7, pp. 91–96, 2023.

[20] R. Wang and I. R. Manchester, “Youla-REN: Learning nonlinear feedback policies with robust stability guarantees,” in *2022 Amer. Control Conf.*, 2022, pp. 2116–2123.

[21] L. Furieri, C. L. Galimberti, M. Zakwan, and G. Ferrari-Trecate, “Distributed neural network control with dependability guarantees: A compositional port-hamiltonian approach,” in *Proc. Learn. Dyn. Control Conf.*, 2022, pp. 571–583.

[22] L. Furieri, C. L. Galimberti, and G. Ferrari-Trecate, “Neural system level synthesis: Learning over all stabilizing policies for nonlinear systems,” in *Proc. 61st Conf. Decis. Control*, 2022, pp. 2765–2770.

[23] N. H. Barbara, R. Wang, and I. R. Manchester, “Learning over contracting and Lipschitz closed-loops for partially-observed nonlinear systems,” in *2023 62nd IEEE Conf. Decis. Control*, 2023, pp. 1028–1033.

[24] C. E. Garcia and M. Morari, “Internal model control. A unifying review and some new results,” *Ind. Eng. Chem. Process Des. Develop.*, vol. 21, no. 2, pp. 308–323, 1982.

[25] C. G. Economou, M. Morari, and B. O. Palsson, “Internal model control: Extension to nonlinear system,” *Ind. Eng. Chem. Process Des. Develop.*, vol. 25, no. 2, pp. 403–411, 1986.

[26] F. Bonassi and R. Scattolini, “Recurrent neural network-based internal model control design for stable nonlinear systems,” *Eur. J. Control*, vol. 65, 2022, Art. no. 100632.

[27] V. Anantharam and C. A. Desoer, “On the stabilization of nonlinear systems,” *IEEE Trans. Autom. Control*, vol. TAC-29, no. 6, pp. 569–572, Jun. 1984.

[28] K. Fujimoto and T. Sugie, “State-space characterization of youla parametrization for nonlinear systems based on input-to-state stability,” in *Proc. 37th Conf. Decis. Control*, 1998, pp. 2479–2484.

[29] K. Fujimoto and T. Sugie, “Characterization of all nonlinear stabilizing controllers via observer-based kernel representations,” *Automatica*, vol. 36, no. 8, pp. 1123–1135, 2000.

[30] D. Ho, “A system level approach to discrete-time nonlinear systems,” in *Proc. Amer. Control Conf.*, 2020, pp. 1625–1630.

[31] K.-K. K. Kim, E. R. Patrón, and R. D. Braatz, “Standard representation and unified stability analysis for dynamic artificial neural network models,” *Neural Netw.*, vol. 98, pp. 251–262, 2018.

[32] M. Revay, R. Wang, and I. R. Manchester, “Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness,” *IEEE Trans. Autom. Control*, vol. 69, no. 5, pp. 2855–2870, May 2024.

[33] Y. Tang, Y. Zheng, and N. Li, “Analysis of the optimization landscape of linear quadratic Gaussian (LQG) control,” in *Proc. Conf. Learn. Dyn. Control*, 2021, pp. 599–610.

[34] L. Furieri and M. Kamgarpour, “First order methods for globally optimal distributed controllers beyond quadratic invariance,” in *Proc. Amer. Control Conf.*, 2020, pp. 4588–4593.

[35] D. E. Rivera, M. Morari, and S. Skogestad, "Internal model control: PID controller design," *Ind. Eng. Chem. Process Des. Develop.*, vol. 25, no. 1, pp. 252–265, 1986.
[36] K. Zhou and J. C. Doyle, *Essentials of Robust Control*, vol. 104. Upper Saddle River, NJ, USA: Prentice Hall, 1998.
[37] M. W. Fisher, G. Hug, and F. Dörfler, "Approximation by simple poles–Part I: Density and geometric convergence rate in hardy space," *IEEE Trans. Autom. Control*, vol. 69, no. 8, pp. 4894–4909, Aug. 2024.
[38] M. W. Fisher, G. Hug, and F. Dörfler, "Approximation by simple poles–Part II: System level synthesis beyond finite impulse response," 2022, *arXiv:2203.16765*.
[39] L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour, "Sparsity invariance for convex design of distributed controllers," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 4, pp. 1836–1847, Dec. 2020.
[40] Y.-S. Wang, N. Matni, and J. C. Doyle, "A system-level approach to controller synthesis," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4079–4093, Oct. 2019.
[41] L. Conger, J. S. L. Li, E. Mazumdar, and S. L. Brunton, "Nonlinear system level synthesis for polynomial dynamical systems," in *Proc. 61st Conf. Decis. Control*, 2022, pp. 3846–3852.
[42] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems Part one: Conditions derived using concepts of loop gain, conicity, and positivity," *IEEE Trans. Autom. Control*, vol. TAC-11, no. 2, pp. 228–238, Apr. 1966.
[43] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," *Foundations Comput. Math.*, vol. 20, no. 4, pp. 633–679, 2020.
[44] Y. Zheng, L. Furieri, M. Kamgarpour, and N. Li, "Sample complexity of linear quadratic Gaussian (LQG) control for output feedback systems," in *Proc. Conf. Learn. Dyn. Control*, 2021, pp. 559–570.
[45] L. Massai, D. Saccani, L. Furieri, and G. Ferrari-Trecate, "Unconstrained learning of networked nonlinear systems via free parametrization of stable interconnected operators," in *Proc. Eur. Control Conf.*, 2024, pp. 651–656.
[46] D. Saccani, L. Massai, L. Furieri, and G. Ferrari-Trecate, "Optimal distributed control with stability guarantees by training a network of neural closed-loop maps," 2024, *arXiv:2404.02820*.
[47] D. Youla, H. Jabr, and J. Bongiorno, "Modern Wiener-Hopf design of optimal controllers–Part II: The multivariable case," *IEEE Trans. Autom. Control*, vol. TAC-21, no. 3, pp. 319–338, Jun. 1976.
[48] L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour, "An input–output parametrization of stabilizing controllers: Amidst youla and system level synthesis," *IEEE Control Syst. Lett.*, vol. 3, no. 4, pp. 1014–1019, Oct. 2019.
[49] Y. Zheng, L. Furieri, M. Kamgarpour, and N. Li, "System-level, input–output and new parameterizations of stabilizing controllers, and their numerical computation," *Automatica*, vol. 140, 2022, Art. no. 110211.
[50] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1467–1476.
[51] M. G. Boroujeni, C. L. Galimberti, A. Krause, and G. Ferrari-Trecate, "A PAC-Bayesian framework for optimal control with stability guarantees," 2024, *arXiv:2403.17790*.
[52] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/
[53] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
[54] D. Martinelli, C. L. Galimberti, I. R. Manchester, L. Furieri, and G. Ferrari-Trecate, "Unconstrained parametrization of dissipative and contracting neural ordinary differential equations," in *Proc. IEEE Conf. Decis. Control*, 2023, pp. 3043–3048.
[55] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 690–701.
[56] M. Zakwan and G. Ferrari-Trecate, "Neural distributed controllers with port-hamiltonian structures," 2024, *arXiv:2403.17785*.
[57] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control, Robotics, Auton. Syst.*, vol. 3, pp. 269–296, 2020.
[58] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, 2021, Art. no. 109597.
[59] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. Eur. Control Conf.*, 2019, pp. 3420–3431.
[60] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," *Robot. Sci. Syst.*, vol. 13, pp. 1–10, 2017.
[61] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[62] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3834–3839.
[63] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *Int. J. Robot. Res.*, vol. 42, no. 6, pp. 356–370, 2023.
[64] D. Bertsekas, *Lessons From AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Belmont, MA, USA: Athena Scientific, 2022.
[65] A. Martin and L. Furieri, "Learning to optimize with convergence guarantees using nonlinear system theory," *IEEE Control Syst. Lett.*, vol. 8, pp. 1355–1360, 2024.
[66] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, "A neural network approach applied to multi-agent optimal control," in *Proc. Eur. Control Conf.*, 2021, pp. 1036–1041.

LUCA FURIERI (Member, IEEE) received the Ph.D. degree in control and optimization from Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland, in 2020. He was a Postdoctoral Researcher with EPFL, Lausanne, Switzerland. He is currently a Swiss National Science Foundation (SNSF) Ambizione Fellow with EPFL, since 2023. His research focuses on learning and optimal control for distributed decision-making and large-scale safety critical applications. He was the recipient of IEEE Transactions on Control of Network Systems Best Paper Award in 2022, European Control Conference Best Paper Award (finalist) in 2019, and American Control Conference O. Hugo Schuck Best Paper Award in 2018 for his papers.

CLARA LUCÍA GALIMBERTI (Member, IEEE) received the degree in electronic engineering from the Universidad Nacional de Rosario, Rosario, Argentina, in 2018. She is currently working toward the Ph.D. degree with the Dependable Control and Decision Group, EPFL, Lausanne, Switzerland. Her research interests include machine learning and control systems.

GIANCARLO FERRARI-TRECATE (Senior Member, IEEE) received the Ph.D. degree in electronic and computer engineering from the Universita' Degli Studi di Pavia, Pavia, Italy, in 1999. In the spring of 1998, he was a Visiting Researcher with the Neural Computing Research Group, University of Birmingham, Birmingham, U.K. In the fall of 1998, he joined the Automatic Control Laboratory, ETH, Zurich, Switzerland, as a Postdoctoral Fellow. He was appointed Oberassistent with ETH, in 2000. In 2002, he joined INRIA, Rocquencourt, France, as a Research Fellow. From March to October 2005, he was a Researcher with the Politecnico di Milano, Milan, Italy. From 2005 to 2016, he was an Associate Professor with the Dipartimento di Ingegneria Industriale e dell'Informazione, Università Degli Studi di Pavia. Since 2016, he has been a Professor with EPFL, Lausanne, Switzerland. His research interests include scalable control, machine learning, microgrids, networked control systems, and hybrid systems. He is the Founder and current Chair of the Swiss chapter of the IEEE Control Systems Society. He is Senior Editor of IEEE Transactions on Control Systems Technology and was on the editorial boards of *Automatica* and *Nonlinear Analysis: Hybrid Systems*.