Neural System Level Synthesis: Learning over All Stabilizing Policies for Nonlinear Systems

Luca Furieri, Clara Lucía Galimberti, and Giancarlo Ferrari-Trecate

Abstract— **We address the problem of designing stabilizing control policies for nonlinear systems in discrete-time, while minimizing an arbitrary cost function. When the system is linear and the cost is convex, the System Level Synthesis (SLS) approach offers an exact solution based on convex programming. Beyond this case, a globally optimal solution cannot be found in a tractable way, in general. In this paper, we develop a parametrization of *all and only* the control policies stabilizing a given time-varying nonlinear system in terms of the combined effect of 1) a strongly stabilizing base controller and 2) a stable SLS operator to be freely designed. Based on this result, we propose a Neural SLS (Neur-SLS) approach guaranteeing closed-loop stability during and after parameter optimization, without requiring any constraints to be satisfied. We exploit recent Deep Neural Network (DNN) models based on Recurrent Equilibrium Networks (RENs) to learn over a rich class of nonlinear stable operators, and demonstrate the effectiveness of the proposed approach in numerical examples.**

I. Introduction

The optimal control of nonlinear systems subject to global asymptotic stability guarantees is one of the most important and challenging problems in control theory. Despite a rich body of research in nonlinear control methods [1], the general Nonlinear Optimal Control (NOC) problem is well understood only when the system dynamics are linear. Classical approaches for dealing with continuous-time and discrete-time NOC include dynamic programming and the associated Bellman optimality principle, and the maximum principle for deterministic problems [2], [3]. Unfortunately, application of these methods to general NOC problems is hindered by methodological and computational challenges [3].

An alternative approach is provided by receding-horizon control schemes such as Nonlinear Model Predictive Control (NMPC) [4]. The NMPC strategy is based on real-time optimization; a finite-horizon open-loop control problem is solved at each time instant for defining the control input at the same time. One of the main drawbacks of NMPC is that the resulting control policy is not available offline in explicit form. Furthermore, several applications may not allow for sufficient computational resources to solve a mathematical program in real-time.

More recently, Reinforcement Learning (RL) and Deep Neural Networks (DNN) have emerged as powerful tools helping agents learn how to make sense of and optimally interact with complex environments [5]. Despite the success of DNN control policies for finite-horizon control tasks [6], [7], RL techniques cannot provide closed-loop stability guarantees in general. As a result, their applicability is limited to systems that are not safety-critical.

Several methods to learn provably stabilizing DNN control policies have been proposed. We divide them into two categories. First, *constrained optimization approaches* [8]–[10] ensure global or local stability by enforcing appropriate Lyapunov-like inequalities during optimization. However, optimizing over conservative stability constraints may excessively limit the set of admissible policies. Further, enforcing constraints such as Linear Matrix Inequalities (LMIs) becomes a computational bottleneck in large-scale applications. Second, *unconstrained optimization approaches* seek to characterize classes of control policies with *built-in* stability guarantees [11], [12]. The unconstrained approach allows one to learn over real-valued parameters through unconstrained gradient descent, without compromising stability for arbitrarily chosen parameter values and whilst enjoying a low computational burden. The main challenge is to characterize stabilizing policy classes that are broad enough to achieve optimized performance. An original perspective on parametrizing stabilizing control policies is based on the nonlinear System Level Synthesis (SLS) [13], which proposes to directly learn over stable nonlinear closed-loop maps — rather than over stabilizing nonlinear control policies. Unfortunately, nonlinear functional equality constraints must be satisfied during optimization [13]. This aspect poses obstacles to practical implementation beyond the well-known linear case [14].

A. Contributions

Motivated by the SLS framework, we establish an unconstrained parametrization of *all* stabilizing nonlinear control policies for nonlinear time-varying systems in discrete-time. Our key finding is that the intractable constraints of nonlinear SLS [13] need not be imposed during optimization, if a stabilizing base control policy is known, and if this base control policy defines itself a stable map from its inputs to its outputs. In this case, all and only the other stabilizing policies are parametrized in terms of a freely chosen stable SLS operator. A main practical benefit is that, no matter how suboptimally this stable SLS operator may be chosen, closed-loop stability is guaranteed by design. Our result extends nonlinear variants of the Youla parametrization [15], [16] to discrete-time system. Further, we recover the recent result of [11] as a subcase, where the authors address the NOC problem for linear systems with nonlinear costs. Last, we propose using deep RENs [17] as a finite-dimensional approximation of nonlinear stable operators, thus extending the well-known idea of Finite Impulse Responses (FIRs) approximations [14] which is valid for linear SLS only. The resulting Neur-SLS optimal control method is tested on formation control problems involving vehicles with nonlinear dynamics and nonlinear costs with collision avoidance terms.

The paper is structured as follows. Section II presents the NOC problem under study, and reviews the nonlinear SLS parametrization of stabilizing control policies. Section III presents our main theoretical results. Section IV introduces deep RENs as an unconstrained finite-dimensional approximation of nonlinear stable operators and formulates the unconstrained Neur-SLS learning problem. In Section V the Neur-SLS method is tested through numerical examples.

Notation: The set of all sequences $\mathbf{x} = (x_0, x_1, x_2, \ldots)$, where $x_t \in \mathbb{R}^n$ for all $t \in \mathbb{N}$ and $\mathbb{N}$ is the set of natural numbers, is denoted as ℓ^n. Moreover, $\mathbf{x}$ belongs to $\ell_p^n \subset \ell^n$ with $p \in \mathbb{N}$ 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 $\sup_t |x_t| < \infty$. 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 $A_t(x_{t:0}) = A_t(0, x_{t-1:0})$, then $\mathbf{A}$ is said to be strictly causal. Similarly, $A_{j:i}(x_{j:0}) = (A_i(x_{i:0}), A_{i+1}(x_{i+1:0}), \ldots, A_j(x_{j:0}))$. For a matrix $M \in \mathbb{R}^{m\times n}$, $M\mathbf{x} = (Mx_0, Mx_1, \ldots) \in \ell^m$. An operator $\mathbf{A} : \ell^n \to \ell^m$ is said to be ℓ_p-stable if it is causal and $\mathbf{A}(\mathbf{a}) \in \ell_p^m$ for all $\mathbf{a} \in \ell_p^n$. Equivalently, we write $\mathbf{A} \in \mathcal{L}_p$.

II. Problem Statement and Nonlinear System Level Synthesis

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 $x_t \in \mathbb{R}^n$ denotes the state vector, $u_t \in \mathbb{R}^m$ denotes the control input and $w_t \in \mathbb{R}^n$ stands for unknown process noise with $w_0 = x_0$. 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 $\mathbf{F}(\mathbf{x}, \mathbf{u}) = (0, f_1(x_0, u_0), \ldots, f_t(x_{t-1:0}, u_{t-1:0}), \ldots)$. Further, we assume that $w_t \sim \mathcal{D}_t(\mu_t, \Sigma_t)$, i.e., w_t is distributed according to an unknown distribution $\mathcal{D}_t$ with expected value μ_t and covariance matrix $\Sigma_t \succ 0$, and that $\mathbf{w}$ belongs to ℓ_p^n.

In order to control the behavior of system (1) we consider nonlinear, dynamic, time-varying feedback control policies

$$\mathbf{u} = \mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \ldots, K_t(x_{t:0}), \ldots)\,, \tag{3}$$

where $\mathbf{K}$ is a causal operator $\mathbf{K} : \ell^n \to \ell^m$ to be designed. It is easy to see that for each sequence of disturbances $\mathbf{w} \in \ell_p^n$ the closed-loop system (1)-(3) produces unique trajectories. Hence, the closed-loop mappings $\mathbf{w} \mapsto (\mathbf{x}, \mathbf{u})$ are well-defined. Specifically, for a system $\mathbf{F}$ and a controller $\mathbf{K}$, we denote the corresponding induced closed-loop maps $\mathbf{w} \mapsto \mathbf{x}$ and $\mathbf{w} \mapsto \mathbf{u}$ as $\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}]$ and $\mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]$, respectively. Therefore, we have $\mathbf{x} = \mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}](\mathbf{w})$ and $\mathbf{u} = \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}](\mathbf{w})$ for all $\mathbf{w} \in \ell_p^n$.

Our goal is to synthesize a control policy $\mathbf{K}$ that complies with two requirements:

(R1) The closed-loop maps $\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}]$ and $\mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]$ are ℓ_p-stable.

(R2) A loss function

$$J = \mathbb{E}_{w_{T:0}}\left[\sum_{t=0}^{T} l(x_t, u_t)\right]\,, \tag{4}$$

is globally minimized, where l is piecewise differentiable and $l(x, u) \geq 0$ for all $(x, u) \in \mathbb{R}^{m+n}$.

Requirement (R1) is a *hard* constraint to be satisfied for all the policies we optimize over. In other words, we require fail-safe learning, in the sense that closed-loop stability must be guaranteed during and after optimization of the parameters describing the policies. Instead, we treat (R2) as a "soft" requirement. We do not expect that a gradient-descent-based algorithm finds 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 [18], [19]. Furthermore, the expected value can only be computed empirically because the distribution of $\mathbf{w}$ is unknown. Last, we note that the theoretical results of Section III also apply to convergent infinite-horizon costs [2]; we focus on the finite-horizon cost (4) to enable the DNN implementation of Section IV.

We are ready to formulate the NOC problem as

$$\begin{aligned} \text{NOC:} \quad & \min_{\mathbf{K}(\cdot)} \quad \mathbb{E}_{w_{T:0}}\left[\sum_{t=0}^{T} l(x_t, u_t)\right] \\ & \text{s. t.} \quad x_t = f_t(x_{t-1:0}, u_{t-1:0}) + w_t\,, \quad w_0 = x_0\,, \\ & \qquad\; u_t = K_t(x_{t:0})\,, \quad \forall t = 0, 1, \ldots\,, \\ & \qquad\; (\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]) \in \mathcal{L}_p\,. \end{aligned}$$

Searching over the space of stabilizing control policies leads to intractable optimization problems in general. The idea behind the SLS approach [13], [14] 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

$$\mathcal{CL}[\mathbf{F}] = \{(\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]) |\ \mathbf{K} \text{ is causal}\}\,, \tag{5}$$

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

$$\mathcal{CL}_p[\mathbf{F}] = \{(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}[\mathbf{F}] |\ (\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{L}_p\}\,. \tag{6}$$

Note that, if $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$, then $\mathbf{x} = \mathbf{\Psi^x}(\mathbf{w}) \in \ell_p^n$ and $\mathbf{u} = \mathbf{\Psi^u}(\mathbf{w}) \in \ell_p^m$ for all $\mathbf{w} \in \ell_p^n$. Based on Theorem III.3 of [13], 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 1 (*Nonlinear SLS parametrization [13]*):*

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

$$\begin{aligned} \mathcal{CL}_p[\mathbf{F}] = \{(\mathbf{\Psi^x}, \mathbf{\Psi^u}) |\ \ & (\mathbf{\Psi^x}, \mathbf{\Psi^u}) \text{ are causal}\,, & (7)\\ & \mathbf{\Psi^x} = \mathbf{F}(\mathbf{\Psi^x}, \mathbf{\Psi^u}) + \mathbf{I}\,, & (8)\\ & (\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{L}_p\}\,. & (9) \end{aligned}$$

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

$$\mathbf{u} = \mathbf{K}(\mathbf{x}) = \mathbf{\Psi^u}((\mathbf{\Psi^x})^{-1}(\mathbf{x}))\,, \tag{10}$$

is unique, and it produces the stable closed-loop responses $(\mathbf{\Psi^x}, \mathbf{\Psi^u})$.

Theorem 1 clarifies that any policy $\mathbf{K}(\mathbf{x})$ achieving ℓ_p-stable closed-loop maps can be described in terms of two operators $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{L}_p$ complying with the nonlinear functional equality (8). Therefore, the NOC problem admits an equivalent Nonlinear SLS (N-SLS) formulation:

$$\begin{aligned} \text{N-SLS:} \quad & \min_{(\mathbf{\Psi^x}, \mathbf{\Psi^u})} \quad \mathbb{E}_{w_{T:0}}\left[\sum_{t=0}^{T} l(x_t, u_t)\right] \\ & \text{s.t.} \quad x_t = \Psi_t^x(w_{t:0})\,, \quad u_t = \Psi_t^u(w_{t:0})\,, \\ & \qquad (\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]\,, \forall t = 0, 1, \dots \end{aligned} \tag{11}$$

According to Theorem 1, the constraint $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$ is equivalent to requiring that $(\mathbf{\Psi^x}, \mathbf{\Psi^u})$ comply with both (8) and (9), which are challenging to satisfy simultaneously. The constraint (8) simply defines the operator $\mathbf{\Psi^x}$ in terms of $\mathbf{\Psi^u}$ and can be computed explicitly because $\mathbf{F}$ is strictly causal. Instead, it is hard to select $\mathbf{\Psi^u} \in \mathcal{L}_p$ such that the corresponding $\mathbf{\Psi^x}$ satisfies $\mathbf{\Psi^x} \in \mathcal{L}_p$. The paper [13] suggests directly searching over ℓ_p-stable operators $(\mathbf{\Psi^x}, \mathbf{\Psi^u})$ and abandoning the goal of complying with (8) exactly. One can then study robust stability when (8) only holds approximately. However, this way of proceeding may result in conservative control policies. In this work, we turn our attention to the problem of complying with (8)-(9) *by design*, thus leading to an unconstrained parametrization of all stabilizing controllers.

III. Main Result: Unconstrained Parametrization of all Stabilizing Controllers

In this section we show that, if *a single* stabilizing control policy $\mathbf{K}'(\mathbf{x})$ with $\mathbf{K}' \in \mathcal{L}_p$ is available for the nonlinear system $\mathbf{F}$, it is possible to parametrize *all other* stabilizing control policies in terms of stable maps $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ by applying the control input

$$\mathbf{u} = \mathbf{K}'(\mathbf{x}) + \boldsymbol{\mathcal{M}}(\mathbf{w})\,, \tag{12}$$

where $\mathbf{w}$ is reconstructed as $\mathbf{x} - \mathbf{F}(\mathbf{x}, \mathbf{u})$. Furthermore, if $\mathbf{K}'$ is stabilizing, but not itself an operator in $\mathcal{L}_p$, the control policy (12) achieves ℓ_p-stable closed-loop maps $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$ for any $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$. From now on, we consider the system

$$\mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w}\,, \quad \mathbf{u} = \mathbf{K}'(\mathbf{x}) + \mathbf{v}\,, \tag{13}$$

$$\mathbf{v} = \boldsymbol{\mathcal{M}}(\mathbf{w})\,, \tag{14}$$

where $\mathbf{K}'$ is the *base controller*, and the operator $\boldsymbol{\mathcal{M}} : \ell^n \to \ell^m$ must be designed in order to comply with the closed-loop stability requirement (R1) and the optimality requirement (R2). The combined effect of $\mathbf{K}'(\mathbf{x})$ and $\boldsymbol{\mathcal{M}}(\mathbf{w})$ defines an overall control policy $\mathbf{K}$ such that $\mathbf{u}(\mathbf{x}) = \mathbf{K}(\mathbf{x})$ is equivalent to (12).

Akin to the Youla parametrization for linear systems [20], the role of a base controller $\mathbf{K}'$ is to appropriately stabilize the system; this allows defining a set of "stable coordinates" and then freely optimize over $\boldsymbol{\mathcal{M}}$.

Definition 1 (ℓ_p-Input-State stabilizing controller): Given $p \in \mathbb{N} \cup \infty$, we say that the base controller $\mathbf{K}'$ is ℓ_p-Input-State (ℓ_p-IS) stabilizing for $\mathbf{F}$ if the map $(\mathbf{w}, \mathbf{v}) \mapsto (\mathbf{x}, \mathbf{u})$ defined by (13) lies in $\mathcal{L}_p$.

We remark that the notion of ℓ_p-IS stabilizability is linked to those of incremental passivity [21] and input-state-stability (ISS) [22] for discrete-time systems. For instance, if a system is ISS-stable, then the control policy $\mathbf{K}' = 0$ is ℓ_∞-IS stabilizing. While this paper does not deal with the computation of a base controller, we refer the interested reader to [21] and references therein for modern methods to design ℓ_p-IS stabilizing controllers in discrete-time. Last, note that our results may be extended to *local* notions of stability, for which a locally stabilizing base controller would be sufficient [16].

Definition 2 (Strongly ℓ_p-IS stabilizing controller): We say that the base controller $\mathbf{K}'$ is *strongly ℓ_p-IS stabilizing* if it is ℓ_p-IS stabilizing, and additionally $\mathbf{K}' \in \mathcal{L}_p$.

We are ready to state our main result.

Theorem 2: Consider the closed-loop system (13)-(14).

1) Assume that $\mathbf{K}'$ is ℓ_p-IS stabilizing. Then, $(\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]) \in \mathcal{CL}_p[\mathbf{F}]$ for every $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$.
2) If, in addition, $\mathbf{K}'$ is strongly ℓ_p-IS stabilizing, then, for any $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$, there exists $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ such that the control policy (12) achieves the closed-loop maps $(\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]) = (\mathbf{\Psi^x}, \mathbf{\Psi^u})$.

Proof: 1) is a restatement of Definition 1 with the substitution $\mathbf{v} = \boldsymbol{\mathcal{M}}(\mathbf{w})$. We prove 2). Let $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$ and choose

$$\boldsymbol{\mathcal{M}} = -\mathbf{K}'(\mathbf{\Psi^x}) + \mathbf{\Psi^u}\,. \tag{15}$$

Note that the composition of operators in $\mathcal{L}_p$ remains in $\mathcal{L}_p$. Then, since $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{L}_p$, we conclude that $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$. It remains to prove that (15) is such that the resulting control policy (12) achieves the closed-loop maps $(\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}]) = (\mathbf{\Psi^x}, \mathbf{\Psi^u})$. We prove this fact by induction. For the inductive step, we assume that, for any $j \in \mathbb{N}$, we have $\Phi_i^x = \Psi_i^x$ and $\Phi_i^u = \Psi_i^u$ for all $i \in \mathbb{N}$ with $0 \leq i \leq j$, where we have dropped the notation $[\mathbf{F}, \mathbf{K}]$ in the interest of readability. Since $(\mathbf{\Phi^x}[\mathbf{F}, \mathbf{K}], \mathbf{\Phi^u}[\mathbf{F}, \mathbf{K}])$ are closed-loop maps induced by $\mathbf{K}$ and $(\mathbf{\Psi^x}, \mathbf{\Psi^u}) \in \mathcal{CL}_p[\mathbf{F}]$, then

$$\Phi_{j+1}^x = F_{j+1}(\Phi_{j:0}^x, \Phi_{j:0}^u) + I,\ \Psi_{j+1}^x = F_{j+1}(\Psi_{j:0}^x, \Psi_{j:0}^u) + I. \tag{16}$$

Then, by (12), (15), (16), and $x_{j+1:0} = \Phi_{j+1:0}^x(w_{j+1:0})$,

$$\begin{aligned} \Phi_{j+1}^u = & K_{j+1}'\left(F_{j+1:0}(\Phi_{j:0}^x, \Phi_{j:0}^u) + I\right) - \\ & - K_{j+1}'\left(F_{j+1:0}(\Psi_{j:0}^x, \Psi_{j:0}^u) + I\right) + \Psi_{j+1}^u\,. \end{aligned} \tag{17}$$

By inductive assumption $\Phi_i^x = \Psi_i^x$ and $\Phi_i^u = \Psi_i^u$ for every $0 \leq i \leq j$. Hence, (17) simplifies to $\Phi_{j+1}^u = \Psi_{j+1}^u$. By (16), $\Phi_{j+1}^x = \Psi_{j+1}^x$. For the base case when $j = 0$, by strict causality of $\mathbf{F}$ and achievability, we have $\Phi_0^x = \Psi_0^x = I$. Then, (12) and (15) imply $\Phi_0^u = K_0'(\Phi_0^x) - K_0'(\Psi_0^x) + \Psi_0^u = \Psi_0^u$. ■

To summarize, Theorem 2 establishes that the nonlinear functional equalities involved in the N-SLS problem (11) can be dropped by tackling the problem in two steps. First, obtain an ℓ_p-IS stabilizing, yet suboptimal control policy $\mathbf{u} = \mathbf{K}'(\mathbf{x})$.

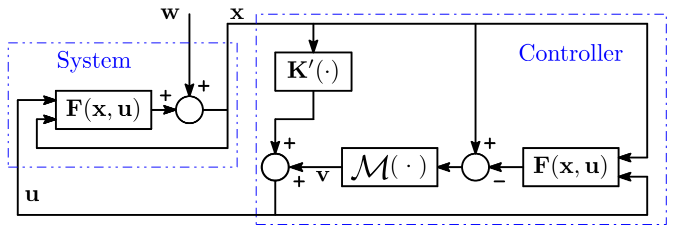
w
x
System
$\mathbf{F}(\mathbf{x}, \mathbf{u})$
+
+
Controller
$\mathbf{K}'(\cdot)$
$\boldsymbol{\mathcal{M}}(\cdot)$
v
+
−
$\mathbf{F}(\mathbf{x}, \mathbf{u})$
u

Fig. 1. Proposed parametrization of all stabilizing controllers in terms of one freely chosen operator $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$.

Second, explore the space of operators $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ such that $\mathbf{u} = \mathbf{K}'(\mathbf{x}) + \boldsymbol{\mathcal{M}}(\mathbf{x} - \mathbf{F}(\mathbf{x}, \mathbf{u}))$ optimizes the closed-loop behavior. The corresponding closed-loop maps will lie in $\mathcal{L}_p$ by design. If $\mathbf{K}'$ is also strongly ℓ_p-IS stabilizing, then, the proposed parametrization is *complete*; all the achievable closed-loop maps — including the ones that are globally optimal for the N-SLS problem (11) — are achieved by appropriately selecting $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ in (12). We illustrate the proposed control architecture in Figure 1.

Remark 1: The assumption of strong stabilizability is pivotal to parametrizing all stabilizing controllers. The classical work [15] has parametrized classes of nonlinear stabilizing controllers in continuous-time based on strongly stabilizing base controllers. The case of linear systems is no exception; a linear strongly stabilizing controller yields a doubly-coprime factorization of the plant, which enables the well-known Youla parametrization [20] of linear stabilizing controllers. From a theoretical point of view, the results of Theorem 2 can be interpreted as the counterpart of [15], [16] for discrete-time control systems, using the lens of SLS.

A. The case of linear systems with nonlinear costs

Consider a linear system of the form

$$x_t = Ax_{t-1} + Bu_{t-1} + w_t\,, \tag{18}$$

and let z denote the time-shift operator. The classical Youla parametrization [20] states that all *linear* stabilizing control policies $\mathbf{u} = \mathbf{K}\mathbf{x}$ can be written in terms of a stable Youla parameter $\mathbf{Q} \in \mathcal{TF}_s$, where $\mathcal{TF}_s$ denotes the set of stable transfer matrices. For the state-feedback case, the Youla approach [20] simplifies to finding a K' such that $(A + BK')$ is Schur-stable, and then expressing all other linear stabilizing control policies as

$$\mathbf{u} = \frac{(K' - \mathbf{Q})}{z}(A\mathbf{x} + B\mathbf{u}) + \mathbf{Q}\mathbf{x}\,, \quad \mathbf{Q} \in \mathcal{TF}_s\,. \tag{19}$$

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 convex-programming. However, these properties are lost when the controller is distributed [23], or the cost function is nonlinear — even if the system dynamics are linear. For the latter scenario, the recent paper [11] characterizes all stabilizing (and contractive) nonlinear policies for output-feedback linear systems by using a nonlinear and contractive Youla parameter. We provide an alternative proof of such result with adaptation to $\mathcal{L}_p$ closed-loop maps in the state-feedback case as an immediate corollary of Theorem 2.

Corollary 1: Consider the linear system (18) and assume that (A, B) is stabilizable. Let K' be such that $(A + BK')$ is Schur-stable. Then, all control policies achieving nonlinear closed-loop maps in $\mathcal{CL}_p$ are expressed as

$$\mathbf{u} = K'\mathbf{x} + \boldsymbol{\mathcal{M}}\left(\mathbf{x} - \frac{(A\mathbf{x} + B\mathbf{u})}{z}\right)\,, \tag{20}$$

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

Proof: The linear base controller K' is ℓ_p-IS stabilizing by assumption and it is strongly stabilizing since $K'\mathbf{x} \in \ell_p^n$ for each $\mathbf{x} \in \ell_p^n$. The conclusion follows by applying point 2 of Theorem 2. ■

Note that, as expected, the linear Youla parametrization (19) is a special case of the proposed parametrization (20) with $\boldsymbol{\mathcal{M}} = \mathbf{Q} - K'$ and $\mathbf{Q} \in \mathcal{TF}_s$.

IV. Neural System Level Synthesis

In this section, we propose the Neur-SLS approach for tackling the NOC problem. Our goal is to exploit Theorem 2 in order to characterize highly expressive DNN control policies which are stabilizing by design — irrespective of how "badly" one may choose the DNN weights. Then, appropriate DNN training leads to optimized performance.

A. Finite-dimensional approximations of $\mathcal{L}_p$ using RENs

An obstacle to directly applying the results of Theorem 2 in practice is that the space $\mathcal{L}_p$ is infinite-dimensional. Hence, a finite-dimensional approximation of $\boldsymbol{\mathcal{M}} \in \mathcal{L}_p$ is necessary for implementation. When linear systems are considered, the SLS approach [14] suggests searching over FIR transfer matrices $\mathbf{M} = \sum_{i=0}^{N} M[i]z^{-i} \in \mathcal{TF}_s$. One can then optimize over the finitely many real matrices $M[i]$ for every $i = 0, \ldots, N$ and obtain near-optimal solutions by selecting the FIR order N to be large enough.

However, the FIR approach limits the search to linear control policies. To parametrize large classes of nonlinear policies, we embrace the recently proposed RENs [17] as finite-dimensional DNN approximations of nonlinear $\mathcal{L}_p$ operators. An operator $\boldsymbol{\mathcal{M}} : \ell^n \to \ell^m$ is a REN if the relationship $\mathbf{u} = \boldsymbol{\mathcal{M}}(\mathbf{w})$ is generated by the following dynamical system:

$$\begin{bmatrix} \xi_t \\ v_t \\ u_t \end{bmatrix} = \overbrace{\begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}}^{W} \begin{bmatrix} \xi_{t-1} \\ \sigma(v_t) \\ w_t \end{bmatrix}\,, \quad \xi_{-1} = 0\,, \tag{21}$$

where $\xi_t \in \mathbb{R}^q$, $v_t \in \mathbb{R}^r$, 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 [17], RENs subsume many existing DNN architectures. In general, RENs define *deep equilibrium network* models [24] due to the implicit relationships between the signals involved in (21). By restricting D_{11} to be strictly lower-triangular, the outputs of (21) can be computed explicitly, thus significantly speeding-up computations [17]. Further, note that the nonlinear relationship defined by (21) is "arbitrarily deep"; the nonlinearity $\sigma(\cdot)$ is recursively applied on the REN internal state ξ_{t-1} and the input w_t for r times, where r is the chosen dimension of v_t.

For an arbitrary choice of W, the map $\boldsymbol{\mathcal{M}}$ induced by (21) may not lie in $\mathcal{L}_p$. The breakthrough of [17] is to provide 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 (21), with the property that the corresponding operator $\boldsymbol{\mathcal{M}}[\theta]$ lies in $\mathcal{L}_2$ by design.[1] We refer to [17] for explicit definition of the mapping $\Theta(\cdot)$.

B. Neur-SLS

By combining the theoretical results of Theorem 2 with REN approximations of operators in $\mathcal{L}_2$, we propose the following unconstrained Neur-SLS optimization problem:

Neur-SLS

$$\min_{\theta\in\mathbb{R}^d} \; \frac{1}{S}\sum_{s=1}^{S}\left[\sum_{t=0}^{T} l(x_t^s, u_t^s)\right] \tag{22}$$

$$\text{s.t.} \quad x_t^s = f_t(x_{t-1:0}^s, u_{t-1:0}^s) + w_t^s\,, \quad w_0^s = x_0^s\,, \tag{23}$$

$$\begin{bmatrix}\xi_t^s\\ v_t^s\\ u_t^s\end{bmatrix} = \begin{bmatrix}0\\ 0\\ K_t'(x_{t:0}^s)\end{bmatrix} + \Theta(\theta)\begin{bmatrix}\xi_{t-1}^s\\ \sigma(v_t^s)\\ x_t^s - f_t(x_{t-1:0}^s, u_{t-1:0}^s)\end{bmatrix}, \tag{24}$$

$$t = 0,1,\ldots\,, \quad \xi_{-1} = 0\,. \tag{25}$$

In the above problem, $\mathbf{K}'$ is an ℓ_2-IS stabilizing base controller, and $\{w_{T:0}^s\}_{s=1}^S$ is a given training set of disturbances. The cost function (22) is defined as the empiric average of the loss evaluated over the training set, and the system dynamics are imposed through (23) for every $\mathbf{w}^s$. The relationships (24)-(25) define a control sequence $\mathbf{u}^s = \mathbf{K}'(\mathbf{x}^s) + \boldsymbol{\mathcal{M}}[\theta](\mathbf{x}^s - \mathbf{F}(\mathbf{x}^s, \mathbf{u}^s))$, where $\boldsymbol{\mathcal{M}}[\theta] \in \mathcal{L}_2$ for every θ. As a result, each value of $\theta \in \mathbb{R}^d$ yields closed-loop maps $(\boldsymbol{\Psi}^{\mathbf{x}}, \boldsymbol{\Psi}^{\mathbf{u}}) \in \mathcal{CL}_2$.

We remark that the class of all ℓ_2-stable REN operators may be significantly more restrictive than the class of all operators in $\mathcal{L}_2$. Hence, learning over all ℓ_2-stable REN operators may not allow reaping the benefits of the completeness result of point 2) of Theorem 2. This is the reason why in the REN-based Neur-SLS implementation (22)-(25) we allow $\mathbf{K}'$ being ℓ_2-IS stabilizing, but not necessarily strongly ℓ_2-IS stabilizing. Based on the above reasoning, an important takeaway of Theorem 2 is that developing finite-dimensional approximations of $\mathcal{L}_p$ that are *as large as possible* is a crucial endeavor — and a worthy one — towards obtaining globally optimal solutions of NOC with the Neur-SLS.

V. Numerical experiments

In this section, we illustrate the use of Neur-SLS to tackle NOC problems. Further, we validate the closed-loop stability guarantees during and after training. We implement the Neur-SLS (22)-(25) using PyTorch and train the resulting DNN with stochastic gradient descent. The implementation details are available in Appendix I. The code to reproduce the examples can be found at `https://github.com/DecodEPFL/neurSLS.git`.

We consider point-mass vehicles with position $p_t \in \mathbb{R}^2$ and velocity $q_t \in \mathbb{R}^2$ subject to nonlinear drag forces (e.g., air or water resistance). The discrete-time model for each vehicle of mass $m \in \mathbb{R}$ is

$$\begin{bmatrix}p_t\\ q_t\end{bmatrix} = \begin{bmatrix}p_{t-1}\\ q_{t-1}\end{bmatrix} + T_s\begin{bmatrix}q_{t-1}\\ m^{-1}\left(-C(q_{t-1})q_{t-1} + F_{t-1}\right)\end{bmatrix}, \tag{26}$$

where $F_t \in \mathbb{R}^2$ denotes the force control input, $T_s > 0$ is the sampling time and $C : \mathbb{R}^2 \to \mathbb{R}$ is a positive *drag function*. Typical scenarios include $C(q) = b|q|_2$ and $C(q) = b$ for some $b > 0$ [25]. By comparing (26) with (1), we remark that the disturbance sequence is given by $((p_0, q_0, 0, 0), (0, 0, 0, 0), \ldots) \in \ell_2^4$. Letting $\overline{p} \in \mathbb{R}^2$ and $\overline{q} = 0_2$ be a target position to be reached with zero velocity, we consider a base controller $u_t = K'(\overline{p} - p_t)$ with $K' = \operatorname{diag}(k_1', k_2')$ and $k_1', k_2' > 0$. Note that the base controller is strongly ℓ_2-IS stabilizing if $C(q) = b$, and strongly ℓ_∞-IS stabilizing if $C(q)$ includes the nonlinearity $b|q|_2$.

We model a set of $N \in \mathbb{N}$ vehicles (26) at time t by defining an overall state $x_t \in \mathbb{R}^{4N}$ and input $u_t \in \mathbb{R}^{2N}$. We select the stage loss function in (22) as

$$l(x_t, u_t) = l_{traj}(x_t, u_t) + l_{ca}(x_t) + l_{obs}(x_t)\,, \tag{27}$$

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 target states and the control energy, $l_{ca}(x_t)$ penalizes collisions between agents and $l_{obs}(x_t)$ penalizes collisions with obstacles in the environment.

A. Results

The scenario `mountains` in Figure 2 involves two agents whose goal is to coordinately pass through a narrow valley. Both agents start from a randomly sampled initial position marked with "$\circ$". The scenario `swapping` in Figure 3 considers twelve agents switching their positions, while avoiding all collisions.[2] Animations are available in our Github repository

We train Neur-SLS control policies to optimize the performance (22)-(27) over a horizon of 5 seconds with sampling time $T_s = 0.05$s, resulting in $T = 100$ time-steps. We consider a linear drag force $C(q)q = bq$ for `swapping` and a nonlinear drag force $C(q)q = b_1 q + b_2|q|_2 q$ for `mountains`, with suitable $b, b_1, b_2 > 0$. The trajectories after training are reported in Figures 2 and 3. The trained Neur-SLS control policies avoid collisions and achieve optimized trajectories thanks to minimizing (27).

Despite training for a finite-horizon, Theorem 2 guarantees closed-loop stability (i.e., ℓ_2-stability for `swapping` and ℓ_∞-stability for `mountains`) around the target positions for $t \to \infty$. To validate this result, in Figure 4 we consider the `mountains` environment and report the cumulative stage loss $\sum_{k=0}^{t} l(x_k^s, u_k^s)$ at $t = 0, 1, \ldots, 500$ for 10 randomly sampled initial conditions $\{w_0^s\}_{s=1}^{10}$ before and after the training. Despite being extremely high before training due to a randomly

[1]Furthermore, RENs enjoy contractivity — although the theoretical results of this paper do not use this property.

[2]The `mountains` and `swapping` benchmarks are motivated by the examples in [6], [12].

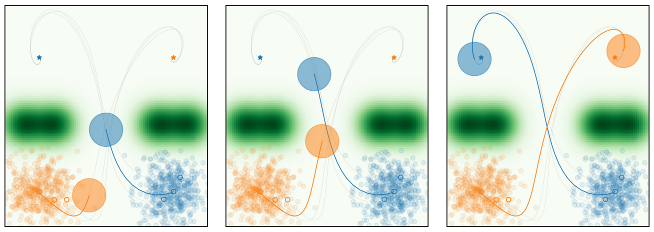

Fig. 2. `Mountains` — Closed-loop trajectories after training over 500 randomly sampled initial conditions marked with ∘. Snapshots taken at times $\tau_1 = 0.7$, $\tau_2 = 1.1$ and $\tau_3 = 5.0$ seconds. Colored (gray) lines show the trajectories in $[0, \tau_i]$ ($[\tau_i, \infty)$). Colored balls (and their radius) represent the agents (and their size for collision avoidance).

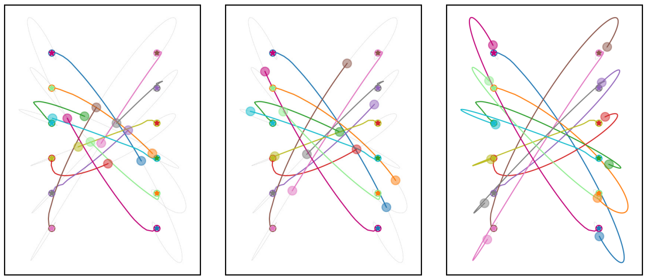

Fig. 3. `Swapping` — Closed-loop trajectories after training. Snapshots taken at times $\tau_1 = 1.25$, $\tau_2 = 1.75$ and $\tau_3 = 5.0$ seconds. Colored (gray) lines show the trajectories in $[0, \tau_i]$ ($[\tau_i, \infty)$). Colored balls (and their radius) represent the agents (and their size for collision avoidance).

selected control policy, the cumulative loss converges to a constant value. Hence, the vehicles reach the target formation with vanishing control inputs as $t \to \infty$ irrespective of the chosen DNN parameters. This fact validates the built-in ℓ_∞-stability of the closed-loop maps.

VI. Conclusions

As we move towards designing deep nonlinear policies for general optimal control problems, it is crucial to guarantee closed-loop stability during and after training. We embrace a system-level perspective and propose a two-step procedure to parametrize all and only the stabilizing controllers in terms of one stable Youla-like operator. This results in a Neur-SLS optimization problem that can be tackled by training a DNN with unconstrained gradient descent. The proposed system-level perspective may open up several venues for future

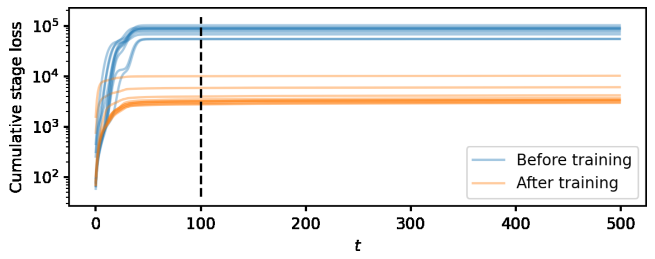

Fig. 4. Cumulative stage loss in `mountains` before and after training. The black dashed line indicates the training horizon $T = 100$.

research, including extension to output-feedback and applications to constrained, distributed and data-driven nonlinear control.

Acknowledgments

We thank Dr. Alessandro Bosso for helpful discussions on stability notions.

References

[1] S. Sastry, *Nonlinear systems: analysis, stability, and control*. Springer Science & Business Media, 2013, vol. 10.
[2] D. P. Bertsekas, “Dynamic programming and optimal control: Vol. I-II,” *Belmont, MA: Athena Scientific*, 2011.
[3] K. Doya, S. Ishii, A. Pouget, and R. P. Rao, *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.
[4] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing, 2017, vol. 2.
[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[6] 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 *IEEE European Control Conference (ECC)*, 2021, pp. 1036–1041.
[7] F. Gama and S. Sojoudi, “Graph neural networks for distributed linear-quadratic control,” in *Learning for Dynamics and Control*. PMLR, 2021, pp. 111–124.
[8] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, “Safe model-based reinforcement learning with stability guarantees,” *Advances in Neural Information Processing Systems 30*, vol. 2, pp. 909–919, 2018.
[9] F. Gu, H. Yin, L. E. Ghaoui, M. Arcak, P. Seiler, and M. Jin, “Recurrent neural network controllers synthesis with stability guarantees for partially observed systems,” *arXiv:2109.03861*, 2021.
[10] P. Pauli, J. Köhler, J. Berberich, A. Koch, and F. Allgöwer, “Offset-free setpoint tracking using neural network controllers,” in *Learning for Dynamics and Control*. PMLR, 2021, pp. 992–1003.
[11] R. Wang, N. Barbara, M. Revay, and I. R. Manchester, “Learning over all stabilizing nonlinear controllers for a partially-observed linear system,” *arXiv:2112.04219*, 2021.
[12] L. Furieri, C. L. Galimberti, M. Zakwan, and G. Ferrari-Trecate, “Distributed neural network control with dependability guarantees: a compositional port-Hamiltonian approach,” *PMLR, to appear*, 2022.
[13] D. Ho, “A system level approach to discrete-time nonlinear systems,” in *IEEE American Control Conference (ACC)*, 2020, pp. 1625–1630.
[14] Y.-S. Wang, N. Matni, and J. C. Doyle, “A system level approach to controller synthesis,” *IEEE Transactions on Automatic Control*, 2019.
[15] V. Anantharam and C. Desoer, “On the stabilization of nonlinear systems,” *IEEE Trans. on Aut. Contr.*, vol. 29, no. 6, pp. 569–572, 1984.
[16] K. Fujimoto and T. Sugie, “State-space characterization of Youla parametrization for nonlinear systems based on input-to-state stability,” in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, vol. 3. IEEE, 1998, pp. 2479–2484.
[17] M. Revay, R. Wang, and I. R. Manchester, “Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness,” *arXiv:2104.05942*, 2021.
[18] Y. Tang, Y. Zheng, and N. Li, “Analysis of the optimization landscape of Linear Quadratic Gaussian (LQG) control,” in *Learning for Dynamics and Control*. PMLR, 2021, pp. 599–610.
[19] L. Furieri and M. Kamgarpour, “First order methods for globally optimal distributed controllers beyond quadratic invariance,” in *IEEE American Control Conference (ACC)*, 2020, pp. 4588–4593.
[20] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.
[21] P. J. Koelewijn, R. Tóth, and S. Weiland, “Incremental dissipativity based control of discrete-time nonlinear systems via the LPV framework,” *arXiv preprint arXiv:2110.00290*, 2021.
[22] Z.-P. Jiang and Y. Wang, “Input-to-state stability for discrete-time nonlinear systems,” *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.
[23] L. Furieri, Y. Zheng, A. Papachristodoulou, and M. Kamgarpour, “Sparsity invariance for convex design of distributed controllers,” *IEEE Trans. on Control of Network Systems*, vol. 7, no. 4, pp. 1836–1847, 2020.
[24] S. Bai, J. Z. Kolter, and V. Koltun, “Deep equilibrium models,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[25] G. Falkovich, *Fluid mechanics: A short course for physicists*. Cambridge University Press, 2011.

Appendix I
Implementation Details

First, we define the addends of the cost function (27). We have

$$l_{traj}(x_t,u_t) = \sum_{i=0}^{N} \left(p_t^i, q_t^i\right)^\intercal \tilde{Q} \left(p_t^i, q_t^i\right) + \alpha_u \left(F_t^i\right)^\intercal \left(F_t^i\right) \,,$$

$$l_{ca}(x_t) = \begin{cases} \alpha_{ca} \sum_{i=0}^{N} \sum_{j,\, i\neq j} (d_{ij,t} + \epsilon)^{-2} & \text{if} \quad d_{ij,t} \leq D \,, \\ 0 & \text{otherwise} \,, \end{cases}$$

where p^i, q^i and F^i denote the position, velocity and control input for agent i, $\tilde{Q} \succeq 0$ and $\alpha_u, \alpha_{ca} \geq 0$ are hyperparameters, $d_{ij,t} = |p_t^i - p_t^j|_2 \geq 0$ denotes the distance between agent i and j and $\epsilon > 0$ is a fixed positive small constant such that the loss remains bounded for all distance values.

Motivated by [6], we represent the obstacles based on a Gaussian density function

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

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

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

Mountains problem (2 robots)

The scenario `mountains` involves two agents whose goal is to coordinately pass through a narrow valley. The system consists of 2 robots (26) of mass $m = 1\,\mathrm{kg}$ and radius $0.5\,\mathrm{m}$, each. We consider a drag function given by

$$C(q)q = b_1 q + b_2 |q| q \,,$$

with $b_1 = 1\,\mathrm{N\,s/m}$ and $b_2 = 0.1\,\mathrm{N\,s/m}$. The based controller has constants $k_1' = k_2' = 1\,\mathrm{N/m}$.

The REN is a deep neural network with depth $r = 32$ layers ($v \in \mathbb{R}^r$). Its internal state ξ is of dimension $q = 32$. We use $\tanh(\cdot)$ as the activation function.

The initial positions of the robots are sampled from a Normal distribution centered at $(\pm 2, -2)$, with covariance $\mathrm{diag}(\sigma^2, \sigma^2)$. We use stochastic gradient descent with Adam in order to minimize the loss function. We set the hyperparameters $\tilde{Q} = \mathrm{diag}(1,1,1,1)$, $\alpha_u = 0.1$, $\alpha_{ca} = 100$ and $\alpha_{obst} = 5000$, and train for 500 epochs with a learning rate of 0.001. We set $\sigma = 0.2$ for the first 300 epochs and then increased it to $\sigma = 0.5$ to further enhance robustness. At each epoch we simulate five trajectories over which we calculate the corresponding loss.

Swapping problem (12 robots)

The scenario `swapping` considers twelve agents switching their positions, while avoiding all collisions. The system consists of 12 robots (26) of mass $m = 1\,\mathrm{kg}$ and radius $0.25\,\mathrm{m}$ each. The considered drag function is given by

$$C(q)q = bq \,,$$

with $b = 1\,\mathrm{N\,s/m}$. The based controller has constants $k_1' = k_2' = 1\,\mathrm{N/m}$.

The REN is a deep neural network with depth $r = 24$ layers ($v \in \mathbb{R}^r$). Its internal state ξ is of dimension $q = 96$. We use $\tanh(\cdot)$ as the activation function.

We use stochastic gradient descent with Adam in order to minimize the loss function. We set the hyperparameters $\tilde{Q} = \mathrm{diag}(1,1,1,1)$, $\alpha_u = 0.1$ and $\alpha_{ca} = 1000$, and train for 1500 epochs with a learning rate of 0.002. Since there are no fixed obstacles in the environment, we set $\alpha_{obst} = 0$.