Proceedings of Machine Learning Research vol vvv:1–24, 2025

Stochastic Neural Simulation Relations for Control Transfer

Alireza Nadali A_NADALI@COLORADO.EDU
Ashutosh Trivedi ASHUTOSH.TRIVEDI@COLORADO.EDU
Majid Zamani MAJID.ZAMANI@COLORADO.EDU

Abstract

This paper explores a neurosymbolic approach to probabilistic transfer of control logic from a source stochastic control system to a target system while ensuring approximately equivalent behavioral guarantees in both domains. Traditional methods struggle with this problem due to the absence of a complete characterization of behavioral specifications, which prevents a direct formulation in terms of loss functions. To address this challenge, we leverage the concept of stochastic simulation relations to establish probabilistic observational equivalence between the behaviors of two stochastic systems. These functions ensure that the outputs of both systems, equipped with their respective controllers, remain probabilistically close over time. By parameterizing stochastic simulation functions with neural networks, we introduce the notion of *stochastic neural simulation functions*, enabling a data-driven mechanism to transfer any synthesized controller—along with its proof of correctness—without requiring explicit specification of behavioral constraints. This neurosymbolic integration combines the scalability of neural methods with the formal guarantees of symbolic approaches, bridging the gap between learning-based control synthesis and formal verification. Compared to prior methods, our approach eliminates the need for a closed-loop mathematical model and explicit requirement specifications for both the source and target systems, while providing probabilistic guarantees over an infinite horizon. We also introduce validity conditions that, when satisfied, ensure the closeness of the outputs of two systems equipped with their corresponding controllers, removing the need for post-facto verification. We demonstrate the effectiveness of our approach through four case studies, highlighting its potential to advance scalable, formally grounded, and transferable control synthesis.

1. Introduction

Symbolic approaches to control design (Rungger and Zamani, 2016) have long been developed for safety-critical systems, where a carefully constructed abstract model enables the formal synthesis of controllers with provable guarantees over the original system. However, constructing such symbolic models demands significant computational effort, posing a major barrier to their widespread adoption. Recently, neural networks have been proposed for controller synthesis, offering various correctness guarantees (Abate et al., 2022). However, these guarantees often require exhaustive state-space exploration, which limits scalability. *Transfer learning* presents a promising alternative for applying neural approaches to control synthesis. By leveraging control logic from a *source system*, it enables the adaptation of controllers to a *target system*, guided by carefully designed loss functions. Since symbolic approaches are computationally feasible for smaller systems, integrating transfer learning with formal guarantees can facilitate an effective, principled, and scalable neurosymbolic approach to control design. In this paper, we propose a general framework for this integration based on stochastic simulation functions.

Transfer Learning. Humans innately exhibit remarkable capabilities in transferring expertise across different tasks, often performing significantly better in one task after learning a related one (Kendler, 1995). *Transfer learning* is a sub-field of artificial intelligence (AI) that focuses on developing similar capabilities for machine learning problems; aimed towards improving learning speed, efficiency, and data requirements. Unlike conventional learning algorithms, which typically focus on individual tasks, transfer learning approaches focus on leveraging knowledge acquired from one or multiple *source* domains to improve learning in a related *target* domain (Weiss et al., 2016). Recently, transfer learning has been successfully applied in designing control logic for dynamical systems (Christiano et al., 2016; Salvato et al., 2021; Nagabandi et al., 2018), albeit without guarantees. However, for safety-critical dynamical systems, control design must provide correctness guarantees, motivating our work. We present a transfer learning approach for stochastic control systems that provides probabilistic guarantees on behavior transfer.

Controller Synthesis and Transfer Learning. This work focuses on controller synthesis for continuous-space stochastic control systems described by difference equations. Examples of such systems include autonomous vehicles, implantable medical devices, and power grids. The safety-critical nature of these systems demands formal guarantees—such as safety, liveness, and more expressive logic-based requirements—on the behavior of the resulting control. While deploying the classic control-theoretic approaches may not necessarily require a mathematical model of the system, and use search and symbolic exploration to synthesize controllers, many of these approaches (Tabuada, 2009) still depend on a mathematical model to provide formal guarantees of correctness. These symbolic approaches typically face the curse-of-dimensionality where the systems with high dimensions become exceedingly cumbersome and time-consuming to design. To overcome these challenges, machine learning based approaches (Zhao et al., 2020; Abate et al., 2022), among others, have been proposed to synthesize control for high-dimensional and complex systems. By making reasonable assumptions (such as Lipschitz continuity) regarding the system, these approaches are able to provide guarantees about their performance. More recently, transfer learning has shown promise (Christiano et al., 2016; Fu et al., 2016; Bousmalis et al., 2018) in transferring controllers from a *source domain* (a *low-fidelity* model or a simulation environment) to a *target domain* (*high-fidelity* model or real system). Some of these approaches (Nadali et al., 2023, 2024a) also aim to transfer *proof certificates* in addition to transferring control.

Specification-Agnostic Control Transfer. Current methodologies (Nadali et al., 2023, 2024a) enable the transfer of control policies and proof certificates when a formal specification is available. However, in typical transfer learning scenarios, control is often inherited from a legacy system deemed desirable for various implicit reasons that are difficult to formalize. As a result, extracting a complete and precise specification becomes challenging. We posit that if structured, unambiguous interfaces—referred to as *semantic anchors* (Velasquez, 2023)—are available to relate observations between the source and target environments, then behaviorally equivalent transfer can be achieved by ensuring the probabilistic closeness of these observations as the system evolves over time. To this end, we introduce *Stochastic Neural Simulation Functions*, which enables the probabilistic transfer of any controller designed for a *source* system to a *target* system, independent of the underlying specification.

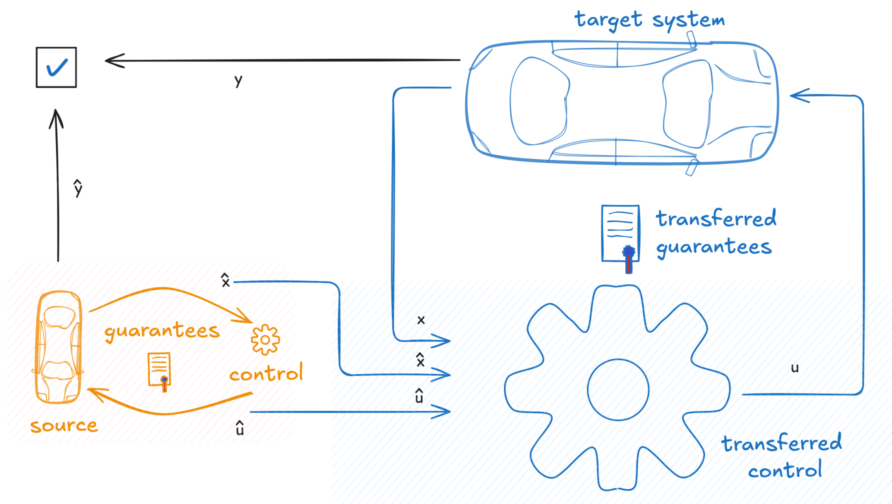


Figure 1: Behavior transfer framework: The existence of a relation and an interface function between source and target implies the closeness of their behaviors.

Stochastic Neural Simulation Functions. For discrete-time stochastic systems with continuous state spaces, finite abstractions were first introduced in (Abate et al., 2008) for the formal synthesis of this class of systems. This method was later refined (Esmaeil Zadeh Soudjani and Abate, 2013) and implemented into `FAUST` (Soudjani et al., 2015). The extension of these techniques to infinite-horizon properties is proposed in (Tkachev and Abate, 2011), while formal abstraction-based policy synthesis is explored in (Tkachev et al., 2013). A novel notion of approximate similarity relation is introduced in (Haesaert and Soudjani, 2020a), accounting for deviations in both stochastic evolution and system outputs. (Lavaei et al., 2019) proposed a method to find an abstraction of networks of stochastic systems.

In this work we assume access to a simulation environment (digital twin or black-box model) of the source system $\hat{\mathfrak{S}}$. In our proposed behavior transfer approach, as depicted in Figure 1, given a source system $\hat{\mathfrak{S}}$ and a target system $\mathfrak{S}$, we design an interface function $\mathcal{K}$ that can transfer an arbitrary controller from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$. It does so by finding a stochastic -approximate- simulation function V between the states of the source and target systems; such that for any pair of related states, and any input in the source environment, there exists an input in the target environment that keeps the next states related according to V. Moreover, it also guarantees that any pair of states, related via V, have similar observations probabilistically. The existence of such simulation functions implies that any behavior on the source system, due to any chosen controller, can be mimicked in the target system. In this work, we train two neural networks to approximate the simulation function V and the interface function $\mathcal{K}$. Under reasonable assumptions, we provide validity conditions that, when satisfied, guarantee the probabilistic lower bound of the outputs of two systems, equipped with their corresponding controllers, thereby eliminating post-facto verification.

Our proposed approach differs from previous work in three key aspects. First, it is model-free, meaning it does not require explicit mathematical equations governing the systems. Second, it provides probabilistic guarantees over an infinite-time horizon. Lastly, we learn an interface function that serves as a controller for the target system—that is, we synthesize a feedback controller rather than focusing solely on verification.

Contribution. This work investigates infinite-horizon output closeness between two given systems. We propose sufficient data-driven criteria, dubbed *Stochastic Neural Simulation Relation*, to ensure probabilistic transfer of controllers designed for source systems, along with their correctness proofs (if existing), to target systems; owing to the explicit computation of the output error bounds related to both systems, this work provides an approach to lift guarantees that is effectively property-independent. In particular, we introduce a training framework that parameterizes the simulation function and its associated interface function as neural networks. Furthermore, by proposing validity conditions to ensure the correctness of these functions, we provide probabilistic guarantees for behavioral transfer from a source to a target system, eliminating the need for post-facto verification.

To the best of our knowledge, this is the first probabilistically correct result that aims to find a stochastic simulation function and its interface function in a data-driven manner between two given systems, for infinite-horizon. In general, existing works are primarily focused on constructing a source (abstract) system given a target (concrete) system (Abate et al., 2022, 2024; Devonport et al., 2021; Hashimoto et al., 2022; Kazemi et al., 2024), deterministic systems (Nadali et al., 2024b), or a fixed specification (Schön et al., 2024). In contrast, our approach does not construct any abstraction. Instead, it establishes a probabilistically correct transfer of controllers designed for a given abstract (source) system to a concrete (target) system. Methods that aim to find a simulation function between two given systems typically make restrictive assumptions about the models of both the source and target systems. For example, the results in (Zhong et al., 2024) assume linear systems, while (Smith et al., 2019) considers only polynomial systems. Furthermore, both methods require access to the mathematical models of the systems. In contrast, our approach makes no assumptions about the specific models of the systems, requiring only access to a black-box representation and the Lipschitz continuity of the dynamics.

Related Work. Transfer learning for control is concerned with transferring a controller from simulation to real-world system which is based on adapting a controller or policy (Fu et al., 2016; Christiano et al., 2016; Bousmalis et al., 2018; Salvato et al., 2021; Nagabandi et al., 2018), or robust control methods that are not affected by the mismatch between the simulator and the real world (Mordatch et al., 2015; Zhou and Doyle, 1998; Berberich et al., 2020). Though these results have shown great promise, they either lack theoretical guarantees or require model of the system. Another approach is to leverage simulation relations (Girard and Pappas, 2011), which is mainly concerned with controlling a complex target system through a simpler source system. (Girard and Pappas, 2011, 2009) introduced a sound hierarchical control scheme based on the notion of an *approximate simulation function (relation)*, bringing together control and automata theory under a unified framework. This relation has had a profound impact on synthesizing controllers against logical properties (da Silva et al., 2019; Fainekos et al., 2007; Zhong et al., 2023) across a variety of systems, such as piecewise affine (Song et al., 2022), control affine (Smith et al., 2019, 2020), and descriptor systems (Haesaert and Soudjani, 2020b). Additionally, it has been applied in various robotics applications, such as legged (Kurtz et al., 2020) and humanoid (Kurtz et al., 2019) robots. Moreover, (Abate et al., 2024) proposed bisimulation learning to find a finite abstract system. The results in (Nadali et al., 2024b) have recently proposed the notion of neural simulation relations for non-stochastic systems. This present work extends that work to handle stochastic systems.

2. Problem Formulation

We denote the set of reals and non-negative reals by $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$, respectively. Given sets A and B, $A \setminus B$ and $A \times B$ represent the set difference and Cartesian product between A and B, respectively, and $|A|$ represents the cardinality of a set A. Moreover, we consider n-dimensional Euclidean space $\mathbb{R}^n$ equipped with infinity norm, defined as $\|x - y\| := \max_{1 \leq i \leq n} |x_i - y_i|$ for $x=(x_1, x_2, \ldots, x_n), y=(y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$. Furthermore, we denote the mean squared loss as $MSE(x, y) := \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2$, where $x, y \in \mathbb{R}^n$. A function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be class κ function if it is continuous, strictly increasing, and $\gamma(0) = 0$. A class κ function is said to be a class κ_∞ function if $\gamma(r) = \infty$ as $r \to \infty$.

Definition 1 (Discrete-Time Stochastic Control System) *A discrete-time stochastic control system (dtSCS) is a tuple $\mathfrak{S}:=(\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$, where $\mathcal{X} \subseteq \mathbb{R}^n$ represents the state set, $\mathcal{X}_0 \subseteq \mathcal{X}$ is the initial state set, $U \subseteq \mathbb{R}^m$ is the set of inputs, and $\mathcal{Y} \subseteq \mathbb{R}^l$ is the set of outputs, V_m is the uncertainty set, and w denotes a sequence of independent and identically distributed (i.i.d.) random variables on the set V_m as $w := \{w(k) : \Omega \to V_m, k \in \mathbb{N}\}$. Furthermore, $f : \mathcal{X} \times U \times V_m \to \mathcal{X}$ is the measurable state transition function, and $h : \mathcal{X} \to \mathcal{Y}$ is the output function. The evolution of the system is described by:*

$$x(t+1) = f(x(t), u(t), w(t)) \text{ and } y(t) = h(x(t)), \quad \text{for all } t \in \mathbb{N}.$$

A state sequence is denoted by $\langle x_0, x_1, \ldots \rangle$, where $x_0 \in \mathcal{X}_0$, and $x(t+1)=f(x(t), u(t), w(t))$, $u(t) \in U$, $w(t) \in V_m$. We assume that sets $\mathcal{X}$, U, and $\mathcal{Y}$ are compact, and maps f and h are unknown but can be simulated via a black-box model. Since the codomain of the map f is $\mathcal{X}$, this implicitly implies that the state set $\mathcal{X}$ is forward invariant, which might seem conservative when dealing with unbounded noise, especially when $\mathcal{X}$ is compact. Following the convention introduced in (Kushner, 1967; Xue, 2024; Anand et al., 2022), to ensure the forward invariance of $\mathcal{X}$, we adopt the standard assumption of stopping the stochastic process. Moreover, we assume that f and h are Lipschitz continuous, as stated in the following assumption.

Assumption 2 (Lipschitz Continuity) *Consider a dtSCS $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$. The map f is Lipschitz continuous in the sense that there exists constants $\mathcal{L}_u, \mathcal{L}_x \in \mathbb{R}_{\geq 0}$ such that for all $x, x' \in \mathcal{X}$, and $u, u' \in U$, one has:*

$$\|f(x, u, w) - f(x', u', w)\| \leq \mathcal{L}_x \|x - x'\| + \mathcal{L}_u \|u - u'\|. \tag{1}$$

Furthermore, the map h is Lipschitz continuous in the sense that there exists a constant $\mathcal{L}_h \in \mathbb{R}_{\geq 0}$ such that for all $x, x' \in \mathcal{X}$, one has $\|h(x) - h(x')\| \leq \mathcal{L}_h \|x - x'\|$.

Without loss of generality, we assume that Lipschitz constants of functions f and h are known. If the Lipschitz constants are unknown, one can leverage sampling methods (Calliess et al., 2020) to estimate those constants.

Definition 3 (Stochastic Behavior Transfer) *Consider two dtSCSs $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$ and $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$, representing the target and the source system, respectively.* A stochastic behavior transfer *from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$ exists if, for any state sequence $\hat{x}(t)$, $\forall t \in \mathbb{N}$, in the source system equipped with its controller, there exists* *a controller and a state sequence $x(t)$, $\forall t \in \mathbb{N}$, in the target system, such that the following holds with confidence $\beta \in (0,1)$:*

$$\mathbb{P}[\max_{t\in\mathbb{N}} \|h(x(t) - \hat{h}(\hat{x}(t))\| \leq \epsilon | x(0), \hat{x}(0)] \geq 1-\gamma,$$

for some $\epsilon, \gamma \in \mathbb{R}_{>0}$.

Intuitively, if a *stochastic behavior transfer* exists from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$, any control policy designed for $\hat{\mathfrak{S}}$ can be adapted to $\mathfrak{S}$ while ensuring that their outputs remain bounded with probability $1-\gamma$ and confidence of β. To automate the transfer of control policies, we pose the following *stochastic behavior transfer* problem.

Problem 4 (Stochastic Behavior Transfer) *Consider two dtSCSs $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_M, w)$ and $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_M, \hat{w})$, representing the target and source systems, respectively. Determine whether a behavior transfer from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$ exists.*

Our solution to the behavior transfer problem (Problem 4) utilizes the following notion.

Definition 5 (Stochastic Approximate Simulation Function) *Consider two dtSCSs $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_M, w)$ and $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$, representing the target system and the source system, respectively. A function $V := \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ is a stochastic approximate simulation function from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$ if following conditions hold for a $\alpha \in \kappa_\infty$:*

$$(i)\ \alpha(\|h(x) - \hat{h}(\hat{x})\|) \leq V(x, \hat{x}),\ \forall x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}, \tag{2}$$

$$(ii)\ \forall x{\in}\mathcal{X}, \forall \hat{x}{\in}\hat{\mathcal{X}}, \forall \hat{u}{\in}\hat{U}, \exists u{\in}U\ s.\ t.\ \mathbb{E}\big[V(f(x,u,w), \hat{f}(\hat{x},\hat{u},\hat{w}))|x,\hat{x},u,\hat{u}\big]{\leq}V(x,\hat{x}). \tag{3}$$

Note that condition (3) tacitly implies the existence of an interface function $\mathcal{K} : \mathcal{X} \times \hat{\mathcal{X}} \times \hat{U} \to U$, as illustrated in Figure 1, which acts as a transferred controller for $\mathfrak{S}$. To demonstrate the merit of the stochastic approximate simulation relation, in comparing the output trajectories of two dtSCSs in a probabilistic setting, we rely on the following proposition; which shows that one can solve Problem 4 by searching for a *stochastic approximate simulation function* from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$ (if existing).

Proposition 6 (Stochastic Simulation Relations and Transfer) *Consider two dtSCSs $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$ and $\hat{\mathfrak{S}}{=}(\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$, representing the target and the source systems, respectively. If there exists a stochastic approximate simulation function from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$ as in Definition 5, then there exists a stochastic behavior transfer from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$.*

The proof can be found in appendix A.1. From this proposition, Problem 4 reduces to the search for a stochastic approximate simulation function V from $\hat{\mathfrak{S}}$ to $\mathfrak{S}$, along with its associated interface function $\mathcal{K}$. To circumvent the need for mathematical models of $\hat{\mathfrak{S}}$ and $\mathfrak{S}$ and to enable the discovery of V through their black-box representations, we learn the function V and the interface function $\mathcal{K}$ as neural networks (Goodfellow et al., 2016).

Definition 7 *A neural network with $k{\in}\mathbb{N}$ layers is a function $F{:}\mathbb{R}^{n_i}{\to}\mathbb{R}^{n_o}$, which computes an output $y_k{\in}\mathbb{R}^{n_o}$ for any input $y_0{\in}\mathbb{R}^{n_i}$ such that $y_j{=}\sigma(W_j y_{j-1}{+}b_j)$, with $j{\in}\{1,\ldots,k\}$, where W_j and b_j are weight matrix and bias vectors, respectively, and σ is the activation function. Additionally, y_{j-1} and y_j are referred to as the input and output of the j-th layer, respectively.*

In this paper, we consider neural networks with ReLU activation function, defined as $\sigma(x) := \max(0, x)$. Such networks describe Lipschitz continuous functions, with Lipschitz constant $\mathcal{L}_F \in \mathbb{R}_{\geq 0}$, in the sense that for all $x'_1, x'_2 \in \mathbb{R}^{n_i}$, one has:

$$\|F(x'_1) - F(x'_2)\| \leq \mathcal{L}_F \|x'_1 - x'_2\|. \tag{4}$$

We obtain an upper bound for Lipschitz constant of a neural network with ReLU activations using spectral norm (Combettes and Pesquet, 2020). Leveraging Proposition 6, we propose a data-driven approach to learn a neural-network-based stochastic approximate simulation relation from a source system $\hat{\mathfrak{S}}$ to a target system $\mathfrak{S}$, thereby addressing Problem 4.

3. Stochastic Neural Simulation Functions

This section explores the training of neural networks to construct a neural simulation function (cf. Definition 8), addressing Problem 4. To this end, we first introduce the construction of the dataset for training these networks. we consider the training set $\mathcal{T} := \mathcal{X} \times \hat{\mathcal{X}}$. Then, to construct the data sets with finitely many points, we cover $\mathcal{T}$ by finitely many disjoint hypercubes $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_M$, by picking a discretization $\mathfrak{e} > 0$, such that:

$$\|\mathfrak{t} - \mathfrak{t}_i\| \leq \frac{\mathfrak{e}}{2}, \text{ for all } \mathfrak{t} \in \mathcal{T}_i, \tag{5}$$

where $\mathfrak{t}_i$ is the center of hypercube $\mathcal{T}_i$, $i \in \{1, \ldots, M\}$. Accordingly, we pick the centers of these hypercubes as sample points, and denote the set of all sample points by $\mathcal{T}_d := \{\mathfrak{t}_1, \ldots, \mathfrak{t}_M\}$. We discretize $\hat{U}$ in the same manner with discretization parameter $\hat{\mathfrak{e}}$, resulting in data sets $\hat{U}_d$. Having these data sets, we can now introduce the notion of *stochastic neural simulation function.*

Definition 8 (Stochastic Neural Simulation Functions) *Consider two dtSCSs* $\mathfrak{S}=(\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$ *and* $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$*, representing the target and the source system, respectively, and neural networks* $V : \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ *and* $\mathcal{K} : \mathcal{X} \times \hat{\mathcal{X}} \times \hat{U} \to U$*. A function* V *is called a* stochastic neural simulation function *from* $\hat{\mathfrak{S}}$ *to* $\mathfrak{S}$ *with the associated interface function* $\mathcal{K}$*, if for all* $(x, \hat{x}) \in \mathcal{T}_d$ *we have:*

$$a)\ \alpha(\|h(x) - \hat{h}(\hat{x})\|) \leq V(x, \hat{x}) - \eta, \tag{6}$$

$$b)\ \forall \hat{u} \in \hat{U}_d,\ \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(x, \mathcal{K}(x, \hat{x}, \hat{u}), w_k), \hat{f}(\hat{x}, \hat{u}, w_j)) \leq V(x, \hat{x}) - \eta - \delta, \tag{7}$$

where $\eta, \delta \in \mathbb{R}_{>0}$ *are some user-defined robustness parameters, and* α *is a class* κ_∞ *function.*

Due to the stochastic nature of systems, we replaced the expectation with empirical mean, and added δ as a robustness parameter to mitigate the error we incur by replacing the expectation with empirical mean. In order to obtain a neural simulation function V, and its associated interface function $\mathcal{K}$, satisfying (6)-(7), we train the network V with loss l:

$$l := MSE(V(x, \hat{x}), \lambda\alpha(\|h(x) - \hat{h}(\hat{x})\|)), \forall (x, \hat{x}) \in \mathcal{T}_d \text{ s.t. } V(x, \hat{x}) < \alpha(\|h(x) - \hat{h}(\hat{x})\|) + \eta, \tag{8}$$

Algorithm 1 Learning Stochastic Neural Simulation Functions

Input: Sets $\mathcal{X}, U, \hat{\mathcal{X}}, \hat{U}$ for target and source systems, respectively, as in Definition 1; discretization parameters $\mathfrak{e}, \hat{\mathfrak{e}}$ for sets $\mathcal{X}, \hat{\mathcal{X}}, \hat{U}$ as in (5); $\mathcal{L}_x$, $\mathcal{L}_u$, $\mathcal{L}_h$, $\mathcal{L}_{\hat{x}}$, $\mathcal{L}_{\hat{u}}$, $\mathcal{L}_{\hat{h}}$ as introduced in Assumption 2; number of simulations for source $\hat{N}$ and target N systems respectively; upper-bound of variance of simulation function M as in Assumption 9, the architecture of the networks V and $\mathcal{K}$ as in Definition 7; a class κ_∞ function α, and a confidence $\beta \in (0, 1)$.
Output: Neural networks V (for the simulation relation as in Definition 8) and $\mathcal{K}$.

1: Construct data sets $\mathcal{T}_d$, and $\hat{U}_d$ according to (5).
2: Initialize networks V and $\mathcal{K}$ (Goodfellow et al., 2016).
3: $\mathcal{L}_V \leftarrow$ Upper bound of Lipschitz constant of V (Combettes and Pesquet, 2020).
4: $\mathcal{L}_K \leftarrow$ Upper bound of Lipschitz constant of $\mathcal{K}$ (Combettes and Pesquet, 2020).
5: **while** *Conditions (6)-(7) and (10)-(12) are not satisfied* **do**
    Construct losses l and l_k according to (8) and (9), respectively
    Train V with loss l.
    Train $\mathcal{K}$ via loss l_k
    $\mathcal{L}_V \leftarrow$ Upper bound of Lipschitz constant of V (Combettes and Pesquet, 2020).
    $\mathcal{L}_K \leftarrow$ Upper bound of Lipschitz constant of $\mathcal{K}$ (Combettes and Pesquet, 2020).
  end
6: Return V, $\mathcal{K}$

where $\lambda > 1$. Additionally, we train the network $\mathcal{K}$ employing the following loss

$$l_k := MSE(h_1(f(x, \mathcal{K}(x, \hat{x}, \hat{u}))), \hat{h}_1(\hat{f}(\hat{x}, \hat{u}))),\ \forall (x, \hat{x}) \in \mathcal{T}_d,\ \forall \hat{u} \in \hat{U}_d, \tag{9}$$

where $h_1 := \sum_{i=1}^{N} h(f(x, \mathcal{K}(x, \hat{x}, \hat{u}), w_i)))$, and $\hat{h}_1 := \sum_{i=1}^{\hat{N}} \hat{h}(\hat{f}(\hat{x}, \hat{u}, \hat{w}_i))$, are empirical means of outputs of target and source systems, respectively. By leveraging l_k, the network $\mathcal{K}$ is trained to produce an input for the target system such that the outputs of the target and source systems remain close at the next time step, regardless of the input provided to the source system. Note that a stochastic neural simulation function, as in Definition 8, is not necessarily a valid stochastic approximate simulation function as in Definition 5. Since neural networks are trained on finitely many data points, out-of-sample guarantees are required to prove correctness. To tackle this issue, we propose the following validity conditions to show that a stochastic neural simulation function satisfies condition (2)-(3) (cf. Theorem 10).

Assumption 9 *Consider two dtSCSs $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$ (a.k.a. source system) and $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$ (a.k.a. target system), and two fully connected neural networks $V : \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ and $\mathcal{K} : \mathcal{X} \times \hat{\mathcal{X}} \times \hat{U} \to U$, with ReLU activations, satisfying (6)-(7). We assume the following validity conditions:*

$$\alpha\big((\mathcal{L}_h + \mathcal{L}_{\hat{h}})\frac{\mathfrak{e}}{2}\big) + \mathcal{L}_V \frac{\mathfrak{e}}{2} - \eta \leq 0, \tag{10}$$

$$N \times \hat{N} \geq \frac{M}{\delta^2 \beta}, \tag{11}$$

$$\mathcal{L}_V\big(\max\big[(\mathcal{L}_x + \mathcal{L}_u \mathcal{L}_K)\frac{\mathfrak{e}}{2}, \mathcal{L}_{\hat{x}}\frac{\mathfrak{e}}{2} + \mathcal{L}_{\hat{u}}\frac{\hat{\mathfrak{e}}}{2}\big] + 1\big) - \eta \leq 0, \tag{12}$$

where $\eta, \delta \in \mathbb{R}_{>0}$ are user-defined parameters as in Definition 8, $\mathcal{T}_d, \hat{U}_d$ are constructed according to (5) with discretization parameter $\mathfrak{e}$. Additionally, $\mathcal{L}_V, \mathcal{L}_h, \mathcal{L}_{\hat{h}}, \mathcal{L}_K$ are Lipschitz constants of $V, h, \hat{h}$, and $\mathcal{K}$, respectively (cf 2 and (4)), and $\mathcal{L}_x, \mathcal{L}_u$ (resp. $\mathcal{L}_{\hat{x}}, \mathcal{L}_{\hat{u}}$) are Lipschitz constants of the target system (resp. the source system), as in Definition 2, and $M \geq Var(V(f(x, \mathcal{K}(x, \hat{x}, \hat{u}), w), \hat{f}(\hat{x}, \hat{u}, \hat{w})))$, for all $x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{U}$, is an upper bound for the variance of V, and $\beta \in (0, 1)$.

The intuition behind Assumption 9 lies in leveraging Lipschitz continuity to provide formal guarantees. Since neural networks are trained on a finite set of data points, it is crucial to establish out-of-sample performance guarantees to ensure overall correctness. Lipschitz continuity enables us to extend guarantees from a finite set of training data to the entire state set. Assumption 4 serves as a condition that facilitates this extension. Specifically, it ensures that if a sample point (used during training) satisfies the stochastic simulation relation conditions, then all points within a neighborhood centered at the sample point with radius $\mathfrak{e}$ also satisfy those conditions. This approach forms the theoretical foundation needed to bridge the gap between finite data and overall correctness across the entire state set. Based on Definition 8 and Assumption 9, Algorithm 1 summarizes the data-driven construction of a stochastic neural simulation relation from the source to the target systems with formal guarantees.

4. Formal Guarantee for Stochastic Neural Simulation Functions

In this section, we propose the main result of our paper. This result shows that a stochastic neural simulation function acquired by using Algorithm 1, conditioned on its termination, is in fact a stochastic approximate simulation function, *i.e.* it satisfies conditions (2)-(3) and therefore can be deployed to solve Problem 4.

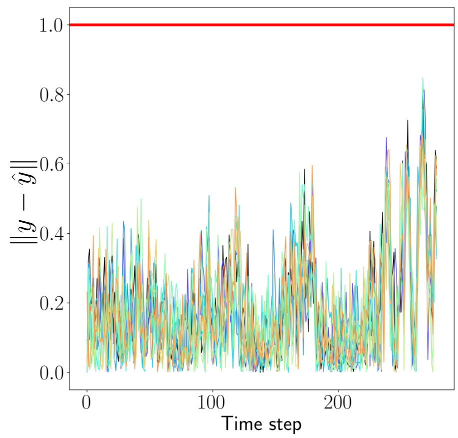
Theorem 10 *Consider two dtSCSs, $\hat{\mathfrak{S}} = (\hat{\mathcal{X}}, \hat{\mathcal{X}}_0, \mathcal{Y}, \hat{U}, \hat{f}, \hat{h}, \hat{V}_m, \hat{w})$ (a.k.a. the source system), with its Lipschitz constants $\mathcal{L}_{\hat{x}}, \mathcal{L}_{\hat{u}}$, and $\mathcal{L}_{\hat{h}}$, and $\mathfrak{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{Y}, U, f, h, V_m, w)$ (a.k.a. the target system), with its Lipschitz constants $\mathcal{L}_x, \mathcal{L}_u$, and $\mathcal{L}_h$. If there exist neural networks V with a Lipschitz constant $\mathcal{L}_V$ and $\mathcal{K}$ with a Lipschitz constant $\mathcal{L}_K$ that satisfy conditions (6) to (12) with κ_∞ function α, then for any closed-loop trajectory of the source system, starting from $\hat{x}_0$, there exists a closed-loop trajectory of the target system equipped with controller $\mathcal{K}$ and starting from x_0 such that with confidence $1 - \beta$, $\beta \in (0, 1)$, the following inequality holds:*

$$\mathbb{P}\big[\max_{t \in \mathbb{N}} \|h(x(t)) - \hat{h}(\hat{x}(t))\| \leq \alpha(\epsilon) | x_0, \hat{x}_0\big] \geq 1 - \frac{V(x_0, \hat{x}_0)}{\alpha(\epsilon)}, \text{ for any } \epsilon \geq 0.$$

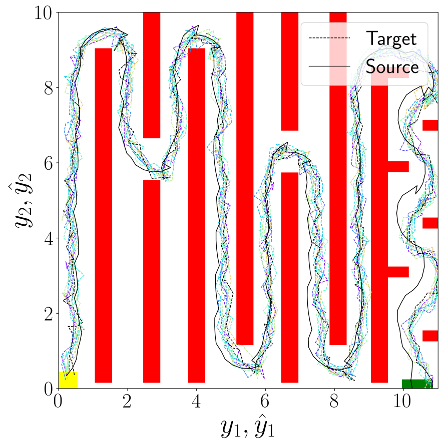
A proof is provided in the appendix A.2. Theorem 10 provides probabilistic closeness of output behaviors of two systems in infinite-horizon with confidence $1 - \beta$.

5. Experimental results

In this paper, the effectiveness of the proposed method is demonstrated through four case studies. We refer the reader to appendix B for the details of all experimental results. In



(a)



(b)

Figure 2: Vehicle control transfer from 3D to 5D. (a) The error between the outputs, and (b) the trajectories for both systems, for 10 different realizations.

particular, we have done a vehicle control transfer from a 3 dimensional model to a 5 dimensional model. The error between outputs of source and target systems over an state sequence of 300 steps is depicted in Figure 2, for 10 different realizations. We leveraged the tool `SCOTS` (Rungger and Zamani, 2016) to design a controller for the source system (without the noise), ensuring it reaches the goal (depicted by the green rectangle) while avoiding obstacles (depicted by red rectangles) from the initial set of states (depicted by the yellow rectangle). In this case study, for $\alpha(\epsilon) = 1$, with 99% confidence, we get: $\mathbb{P}\big[\max_{t\in\mathbb{N}} \|h(x(t)) - \hat{h}(\hat{x}(t))\| \leq 1 | x_0, \hat{x}_0\big] \geq 0.9287$. We conducted these experiments with 10000 different realizations, and in only 52 cases did the difference between the outputs exceed 1, which aligns with the theoretical results.

6. Conclusion

This paper presents a data-driven approach for behavior transfer between a source and target stochastic control systems, offering probabilistic guarantees. We employ neural networks to encode and search for a stochastic simulation function and its corresponding interface function, collectively termed stochastic neural simulation functions. The existence of these functions ensures that the output error between the two systems remains within a bounded range, facilitating probabilistic behavior transfer. To guarantee correctness, we propose validity conditions for the neural networks representing the stochastic simulation and interface functions, eliminating the need for post-facto verification. Experimental results from four case studies demonstrate the effectiveness of the proposed control transfer approach. A promising future direction is to reduce sample complexity by leveraging structural properties of both the source and target systems, such as monotonicity (Angeli and Sontag, 2003) and mixed-monotonicity (Coogan and Arcak, 2015).

References

Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44 (11):2724–2734, 2008.

Alessandro Abate, Alec Edwards, and Mirco Giacobbe. Neural abstractions. *Advances in Neural Information Processing Systems*, 35:26432–26447, 2022.

Alessandro Abate, Mirco Giacobbe, and Yannik Schnitzer. Bisimulation learning. In *International Conference on Computer Aided Verification*, pages 161–183. Springer, 2024.

Daniel Ajeleye, Abolfazl Lavaei, and Majid Zamani. Data-driven controller synthesis via finite abstractions with formal guarantees. *IEEE Control Systems Letters*, 7:3453–3458, 2023.

Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

Mahathi Anand, Vishnu Murali, Ashutosh Trivedi, and Majid Zamani. K-inductive barrier certificates for stochastic systems. In *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2022.

David Angeli and Eduardo D Sontag. Monotone control systems. *IEEE Transactions on automatic control*, 48(10):1684–1698, 2003.

Julian Berberich, Johannes Köhler, Matthias A Müller, and Frank Allgöwer. Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, 66(4):1702–1717, 2020.

Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

Jan-Peter Calliess, Stephen J Roberts, Carl Edward Rasmussen, and Jan Maciejowski. Lazily adapted constant kinky inference for nonparametric regression and model-reference adaptive control. *Automatica*, 122:109216, 2020.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Patrick L Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM Journal on Mathematics of Data Science*, 2(2):529–557, 2020.

Samuel Coogan and Murat Arcak. Efficient finite abstraction of mixed monotone systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 58–67, 2015.

Rafael Rodrigues da Silva, Vince Kurtz, and Hai Lin. Active perception and control from temporal logic specifications. *IEEE Control Systems Letters*, 3(4):1068–1073, 2019.

Alex Devonport, Adnane Saoud, and Murat Arcak. Symbolic abstractions from data: A pac learning approach. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 599–604. IEEE, 2021.

Sadegh Esmaeil Zadeh Soudjani and Alessandro Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, 2013.

Georgios E Fainekos, Antoine Girard, and George J Pappas. Hierarchical synthesis of hybrid controllers from temporal logic specifications. In *Hybrid Systems: Computation and Control: 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007. Proceedings 10*, pages 203–216. Springer, 2007.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. in 2016 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026, 2016.

Antoine Girard and George J Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45(2):566–571, 2009.

Antoine Girard and George J Pappas. Approximate bisimulation: A bridge between computer science and control theory. *European Journal of Control*, 17(5-6):568–578, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511, 2020a.

Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511, 2020b.

Kazumune Hashimoto, Adnane Saoud, Masako Kishida, Toshimitsu Ushio, and Dimos V Dimarogonas. Learning-based symbolic abstractions for nonlinear control systems. *Automatica*, 146:110646, 2022.

MA Hernández. Chebyshev's approximation algorithms and applications. *Computers & Mathematics with Applications*, 41(3-4):433–445, 2001.

Milad Kazemi, Rupak Majumdar, Mahmoud Salamati, Sadegh Soudjani, and Ben Wooding. Data-driven abstraction-based control synthesis. *Nonlinear Analysis: Hybrid Systems*, 52: 101467, 2024.

Tracy S Kendler. *Levels of cognitive development*. Psychology Press, 1995.

Vince Kurtz, Rafael Rodrigues da Silva, Patrick M Wensing, and Hai Lin. Formal connections between template and anchor models via approximate simulation. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 64–71. IEEE, 2019.

Vince Kurtz, Patrick M Wensing, and Hai Lin. Approximate simulation for template-based whole-body control. *IEEE Robotics and Automation Letters*, 6(2):558–565, 2020.

Harold Joseph Kushner. *Stochastic Stability and Control*. Mathematics in Science and Engineering. Academic Press, 1967. ISBN 9780080955407.

Abolfazl Lavaei, Sadegh Soudjani, and Majid Zamani. Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107:125–137, 2019.

Pierre-Jean Meyer, He Yin, Astrid H Brodtkorb, Murat Arcak, and Asgeir J Sørensen. Continuous and discrete abstractions for planning, applied to ship docking. *IFAC-PapersOnLine*, 53(2):1831–1836, 2020.

Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5307–5314. IEEE, 2015.

Alireza Nadali, Ashutosh Trivedi, and Majid Zamani. Transfer learning for barrier certificates. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 8000–8005. IEEE, 2023.

Alireza Nadali, Ashutosh Trivedi, and Majid Zamani. Transfer of Safety Controllers Through Learning Deep Inverse Dynamics Model. In *The 8th IFAC Conference on Analysis and Design of Hybrid Systems, to appear*, 2024a.

Alireza Nadali, Bingzhuo Zhong, Ashutosh Trivedi, and Majid Zamani. Transfer learning for control systems via neural simulation relations. *arXiv preprint arXiv:2412.01783*, 2024b.

Alireza Nadali, Ashutosh Trivedi, and Majid Zamani. On choice of loss functions for neural control barrier certificates, 2025. URL https://openreview.net/forum?id=GFapl0jE7E.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Matthias Rungger and Majid Zamani. Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th international conference on hybrid systems: Computation and control*, pages 99–104, 2016.

Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

Oliver Schön, Birgit van Huijgevoort, Sofie Haesaert, and Sadegh Soudjani. Bayesian formal synthesis of unknown systems via robust simulation relations. *IEEE Transactions on Automatic Control*, 2024.

Stanley W Smith, He Yin, and Murat Arcak. Continuous abstraction of nonlinear systems using sum-of-squares programming. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 8093–8098. IEEE, 2019.

Stanley W Smith, Murat Arcak, and Majid Zamani. Approximate abstractions of control systems with an application to aggregation. *Automatica*, 119:109065, 2020.

Zihao Song, Vince Kurtz, Shirantha Welikala, Panos J Antsaklis, and Hai Lin. Robust approximate simulation for hierarchical control of piecewise affine systems under bounded disturbances. In *2022 American Control Conference (ACC)*, pages 1543–1548. IEEE, 2022.

Sadegh Esmaeil Zadeh Soudjani, Caspar Gevaerts, and Alessandro Abate. Faust: F ormal a bstractions of u ncountable-st ate st ochastic processes. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 272–286. Springer, 2015.

Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

Ilya Tkachev and Alessandro Abate. On infinite-horizon probabilistic properties and stochastic bisimulation functions. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 526–531. IEEE, 2011.

Ilya Tkachev, Alexandru Mereacre, Joost-Pieter Katoen, and Alessandro Abate. Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 293–302, 2013.

Alvaro Velasquez. Transfer from imprecise and abstract models to autonomous technologies (tiamat). *Defense Advanced Research Projects Agency (DARPA) Program Solicitation*, 2023.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

Bai Xue. Sufficient and necessary barrier-like conditions for safety and reach-avoid verification of stochastic discrete-time systems. *arXiv preprint arXiv:2408.15572*, 2024.

Hongchao Zhang, Junlin Wu, Yevgeniy Vorobeychik, and Andrew Clark. Exact verification of relu neural control barrier functions. *Advances in neural information processing systems*, 36:5685–5705, 2023.

Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pages 1–11, 2020.

Bingzhou Zhong, Murat Arcak, and Majid Zamani. Hierarchical control for cyber-physical systems via general approximate alternating simulation relations. In *The 8th IFAC Conference on Analysis and Design of Hybrid Systems*, 2024.

Bingzhuo Zhong, Abolfazl Lavaei, Majid Zamani, and Marco Caccamo. Automata-based controller synthesis for stochastic systems: A game framework via approximate probabilistic relations. *Automatica*, 147:110696, 2023.

Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

Appendix A. Omitted Proofs

A.1. Proof of Proposition 6

Proof Condition (3) implies that process $V(x(t),\hat{x}(t))$ is a nonnegative supermartingle. Therefore, one obtains:

$$\mathbb{P}\big[\max_{t\in\mathbb{N}}\|h(x(t))-\hat{h}(\hat{x}(t))\|\geq\epsilon|x(0),\hat{x}(0)\big] = \mathbb{P}\big[\max_{t\in\mathbb{N}}\alpha(\|h(x(t))-\hat{h}(\hat{x}(t))\|)\geq\alpha(\epsilon)|x(0),\hat{x}(0)\big]$$
$$\leq\mathbb{P}\big[\max_{t\in\mathbb{N}}V(x(t),\hat{x}(t))\geq\alpha(\epsilon)|x(0),\hat{x}(0)] \tag{13}$$
$$\leq\frac{V(x(0),\hat{x}(0))}{\alpha(\epsilon)}, \tag{14}$$

where (14) follows from the nonnegative supermartingle property ((Kushner, 1967), Theorem 12, p. 71), and (13) is obtained by using inequality (2). Therefore:

$$\mathbb{P}\big[\max_{t\in\mathbb{N}}\|h(x(t))-\hat{h}(\hat{x}(t))\|\leq\epsilon|x(0),\hat{x}(0)\big]\geq 1-\frac{V(x(0),\hat{x}(0))}{\alpha(\epsilon)} \tag{15}$$

■

A.2. Proof of Theorem 10

Proof For all $x\in\mathcal{X}$, all $\hat{x}\in\hat{\mathcal{X}}$, all $\hat{u}\in\hat{U}$, consider the following:

$$\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})}V(f(\bar{X}_k),\hat{f}(\hat{X}_j))-V(x,\hat{x}),$$

where $\bar{X}_k:=(x,\mathcal{K}(x,\hat{x},\hat{u}),w_k)$ and $\hat{X}_j:=(\hat{x},\hat{u},\hat{w}_j)$. According to (5), there exists $(x_i,\hat{x}_i)\in\mathcal{T}_d$ and $\hat{u}_i\in\hat{U}_d$ such that $\|(x_i,\hat{x}_i)-(x,\hat{x})\|\leq\frac{\mathfrak{e}}{2}$, and $\|\hat{u}-\hat{u}_i\|\leq\frac{\mathfrak{e}}{2}$, respectively. Then:

$$\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})}V(f(\bar{X}_k),\hat{f}(\hat{X}_j))-V(x,\hat{x})=\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})}V(f(\bar{X}_k),\hat{f}(\hat{X}_j))-V(x,\hat{x})$$
$$-\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})}V(f(\bar{X}_{k,i}),\hat{f}(\hat{X}_{j,i}))+\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})}V(f(\bar{X}_{k,i}),\hat{f}(\hat{X}_{j,i})),$$

where $\bar{X}_{k,i} := (x_i, \mathcal{K}(x_i, \hat{x}_i, \hat{u}_i), w_k)$ and $\hat{X}_{j,i} := (\hat{x}_i, \hat{u}_i, \hat{w}_j)$. Employing Lipschitz continuity of V as in (4), one gets:

$$
\begin{aligned}
&\frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k), \hat{f}(\hat{X}_j)) - V(x, \hat{x}) \\
&\quad - \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) \\
\leq& \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} \mathcal{L}_V \|(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - (f(\bar{X}_k), \hat{f}(\hat{X}_j))\| \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}) \\
\leq& \mathcal{L}_V \|(f(\bar{X}_i), \hat{f}(\hat{X}_i)) - (f(\bar{X}), \hat{f}(\hat{X}))\| \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}) \\
\leq& \mathcal{L}_V \| \max\big[\|f(\bar{X}_i) - f(\bar{X})\|, \|\hat{f}(\hat{X}_i) - \hat{f}(\hat{X}))\|\big] \| \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}) \\
\leq& \mathcal{L}_V \| \max\big[(\|\mathcal{L}_x \|x - x_i\| + \mathcal{L}_u \mathcal{L}_K \|x - x_i\|\|), (\|\mathcal{L}_{\hat{x}} \|\hat{x} - \hat{x}_i\| + \mathcal{L}_{\hat{u}} \|\hat{u} - \hat{u}_i\|\|)\big] \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}) \qquad (16) \\
\leq& \mathcal{L}_V \frac{\mathfrak{e}}{2}\big(\max\big[(\mathcal{L}_x + \mathcal{L}_u \mathcal{L}_K)\frac{\mathfrak{e}}{2}, \mathcal{L}_{\hat{x}} \frac{\mathfrak{e}}{2} + \mathcal{L}_{\hat{u}} \frac{\hat{\mathfrak{e}}}{2}\big]\big) \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}), \\
\leq& \mathcal{L}_V \frac{\mathfrak{e}}{2}\big(\max\big[(\mathcal{L}_x + \mathcal{L}_u \mathcal{L}_K)\frac{\mathfrak{e}}{2}, \mathcal{L}_{\hat{x}} \frac{\mathfrak{e}}{2} + \mathcal{L}_{\hat{u}} \frac{\hat{\mathfrak{e}}}{2}\big]\big) \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) - V(x, \hat{x}) + V(x_i, \hat{x}_i) - V(x_i, \hat{x}_i), \\
\leq& \mathcal{L}_V \frac{\mathfrak{e}}{2}\big(\max\big[(\mathcal{L}_x + \mathcal{L}_u \mathcal{L}_K)\frac{\mathfrak{e}}{2}, \mathcal{L}_{\hat{x}} \frac{\mathfrak{e}}{2} + \mathcal{L}_{\hat{u}} \frac{\hat{\mathfrak{e}}}{2}\big]\big) \\
&\quad + \frac{1}{N \times \hat{N}} \sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_{k,i}), \hat{f}(\hat{X}_{j,i})) + \mathcal{L}_V \frac{\mathfrak{e}}{2} - V(x_i, \hat{x}_i) \qquad (17)
\end{aligned}
$$

where (16) follows from Lipschitz continuity of source and target systems as defined in (1), and (17) follows from Lipschitz continuity of V, respectively. Substituting (7), one gets:

$$\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k),\hat{f}(\hat{X}_j)) - V(x,\hat{x}) \tag{18}$$

$$\begin{aligned}&\leq \mathcal{L}_V\frac{\mathfrak{e}}{2}\big(\max\big[(\mathcal{L}_x+\mathcal{L}_u\mathcal{L}_K)\frac{\mathfrak{e}}{2},\mathcal{L}_{\hat{x}}\frac{\mathfrak{e}}{2}+\mathcal{L}_{\hat{u}}\frac{\hat{\mathfrak{e}}}{2}\big]\big)+\mathcal{L}_V\frac{\mathfrak{e}}{2}-\eta-\delta \overset{(12)}{\to}\\ &\leq -\delta,\quad \text{for all } x\in\mathcal{X},\ \text{all } \hat{x}\in\hat{\mathcal{X}},\ \text{all } \hat{u}\in\hat{U}.\end{aligned} \tag{19}$$

One could use similar argument to show condition (6) along with validity condition (10) implies condition (2), however, it is omitted here for brevity.

As mentioned previously, to train neural networks V and $\mathcal{K}$, we have replaced the expectation with average mean. To capture the error introduced by this, we have added another robustness parameter δ in Definition 8. We utilize Chebyshev's inequality (Hernández, 2001) to quantify such an error with the associated confidence. The difference between empirical mean in (7) and the expected value in (3) can be quantified by invoking the Chebyshev's inequality as:

$$\begin{aligned}&\mathbb{P}_w\Bigg(|\mathbb{E}[V(f(x,\mathcal{K}(x,\hat{x},\hat{u}),w),\hat{f}(\hat{x},\hat{u}),\hat{w})|x,\hat{x}]\\ &-\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k),\hat{f}(\hat{X}_j))|\leq\delta\Bigg)\geq 1-\frac{M}{\delta^2 N\times\hat{N}},\end{aligned} \tag{20}$$

for all $x\in\mathcal{X},\hat{x}\in\hat{\mathcal{X}},\hat{u}\in\hat{U}$, where M is an upper-bound for variance of function V. in which we have $\beta\geq\frac{M}{\delta^2 N\times\hat{N}}$. This implies $N\times\hat{N}\geq\frac{M}{\delta^2\beta}$, which is satisfied according to (11). Finally, consider the following:

$$\begin{aligned}&\mathbb{E}[V(f(x,\mathcal{K}(x,\hat{x},\hat{u}),w),\hat{f}(\hat{x},\hat{u}),\hat{w})|x,\hat{x}]-V(x,\hat{x})=\\ &\mathbb{E}[V(f(x,\mathcal{K}(x,\hat{x},\hat{u}),w),\hat{f}(\hat{x},\hat{u}),\hat{w})|x,\hat{x}]-V(x,\hat{x})\\ &+\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k),\hat{f}(\hat{X}_j))-\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k),\hat{f}(\hat{X}_j))\\ \leq&\mathbb{E}[V(f(x,\mathcal{K}(x,\hat{x},\hat{u}),w),\hat{f}(\hat{x},\hat{u}),\hat{w})|x,\hat{x}]-\frac{1}{N\times\hat{N}}\sum_{(k,j)=(1,1)}^{(N,\hat{N})} V(f(\bar{X}_k),\hat{f}(\hat{X}_j))-\delta,\end{aligned} \tag{21}$$

for all $x\in\mathcal{X}$, all $\hat{x}\in\hat{\mathcal{X}}$, all $\hat{u}\in\hat{U}$, where (21) is followed by (19). According to (20), with probability $1-\beta$, the difference between absolute value of expectation and average mean in (21) is less than δ, thus, with confidence $1-\beta$, neural network V along with its corresponding interface function $\mathcal{K}$ satisfies condition (3), and they form a stochastic approximate simulation function as defined in Definition 5.

■

Appendix B. Experiments

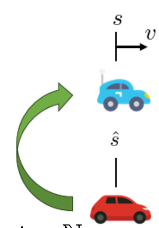
In this section, the effectiveness of the proposed method is demonstrated through four case studies. All experiments are conducted on an Nvidia RTX 4090 GPU. Both networks are parameterized with 5 hidden laters, each containing 200 neurons, and employ ReLU activation. For all experiments, the networks are trained using Algorithm 1 with $\beta = 0.01$. Although mathematical models of all systems are reported for simulation purposes, they were not used to encode neural simulation relation conditions.

B.1. Transfer of Vehicle Control

In these case studies, we aim to transfer a controller designed for a lower-dimensional vehicle model to higher-dimensional ones. The first case study examines the transfer of a simple proportional controller from a one-dimensional vehicle model (position) to a two-dimensional model (position and velocity). The second case study extends this approach by transferring control from a three-dimensional model (with formal guarantees) to a five-dimensional one.

From 1d to 2d. The source system is a one dimensional model:

$$\hat{s}(t+1) = \hat{s}(t) + \tau\hat{u}(t) + \hat{w}(t), \quad \hat{y}(t) = \hat{s}(t), \quad t \in \mathbb{N},$$



where $\hat{s}(t)$ is position and $\hat{u}(t)$ is velocity at time step t. The target system is a 2 dimensional car model:

$$x(t+1) = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0.5\tau^2 \\ \tau \end{bmatrix} u(t) + w(t), \quad y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \quad t \in \mathbb{N},$$

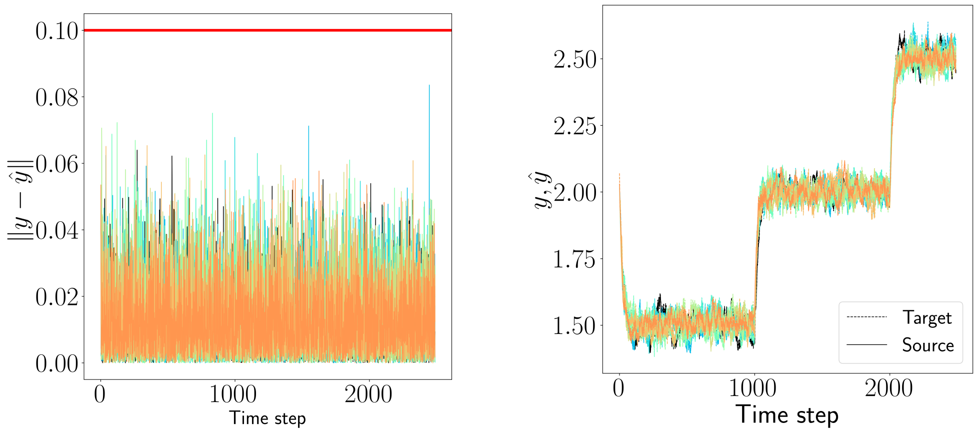
where $\tau = 0.1$ is the sampling time, and $x(t) := [s(t); v(t)]$ is the state vector, in which $s(t)$ and $v(t)$ are position and velocity of the vehicle at time step t, respectively, and $u(t)$ is the acceleration of the vehicle as the control input. Furthermore, we consider $\mathcal{X} = \mathcal{X}_0 = [0, 4] \times [-0.3, 0.3]$, $U = [-0.5, 0.5]$, $\hat{U} = [-0.2, 0.2]$, and $\hat{\mathcal{X}} = \hat{\mathcal{X}}_0 = [0, 4]$ represent the state, initial state and input set of the target system, input and state, and initial state set of the source system, respectively. The corresponding Lipschitz constants are $\mathcal{L}_x = 1.1, \mathcal{L}_u = 0.1, \mathcal{L}_h = 1, \mathcal{L}_{\hat{x}} = 1, \mathcal{L}_{\hat{u}} = 0.1$, and $\mathcal{L}_{\hat{h}} = 1$. Our method converged in 4 minutes with the following parameters: η=0.001, δ=0.048, $\hat{\mathfrak{e}} = \mathfrak{e}$=0.01, $\mathcal{L}_V$=0.5, $N = \hat{N} = 100, M = 0.01, \alpha(x) = \log(1+x)$, and $\mathcal{L}_K$=6.75×10^{-5}. In this case study, for $\alpha(\epsilon) = 0.1$, with 99% confidence, we get:

$$\mathbb{P}\big[\max_{t\in\mathbb{N}} \|h(x(t)) - \hat{h}(\hat{x}(t))\| \leq 0.1 | x_0, \hat{x}_0\big] \geq 0.9467,$$

for $\|h(x(0)) - \hat{h}(\hat{x}(0))\| \leq 0.01$

The error between outputs of source and target systems over an state sequence of 2500 steps is depicted in Figure 3, for 10 different realizations. Source system is controlled by a simple proportional controller, and the setpoint was changed with every 1000 steps. We conducted these experiments with 10000 different realizations, and in only three cases did the difference between the outputs exceed 0.1, which aligns with the theoretical results.

From 3d to 5d. For our next case study, we borrowed the source vehicle model from (Ajeleye et al., 2023), and the target system from (Althoff et al., 2017). Here the target system is



(a) (b)

Figure 3: Vehicle control transfer from 1D to 2D. (a) The error between the outputs, and (b) the trajectories for both systems, for 10 different realizations.

complex five-dimensional model, while the source system is the unicycle model. The target system is a 5 dimensional car:

$$x(t+1) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \delta(t) \\ v(t) \\ \psi(t) \end{bmatrix} + \tau \begin{bmatrix} v(t)\sin(\psi(t)) \\ v(t)\cos(\psi(t)) \\ u_1(t) \\ u_2(t) \\ v(t)\tan(\delta(t)) \end{bmatrix}, y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} x(t),$$

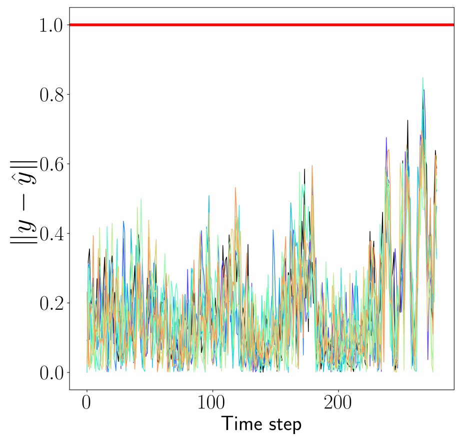
where $\tau = 0.25$ is the sampling time, and $x(t) := [x_1(t); x_2(t); \delta(t); v(t); \psi(t)]$ is the state vector, in which $x_1(t), x_2(t), \delta(t), v(t), \psi(t)$ are horizontal position, vertical position, steering angle, velocity, and heading angle at time step k, respectively. $u_1(t), u_2(t) \in [-1, 1]$ are acceleration and steering of the vehicle as control inputs, at time step t, respectively. The source system is a popular (Zhang et al., 2023; Zhao et al., 2020; Ajeleye et al., 2023) three dimensional unicycle model, given as:

$$\hat{x}(t+1) = \begin{bmatrix} \hat{x}_1(t) \\ \hat{x}_2(t) \\ \hat{x}_3(t) \end{bmatrix} + \tau \begin{bmatrix} \hat{u}_1(t)\cos(q(t) + \hat{x}_3(t))/\cos(q(t)) \\ \hat{u}_1(t)\sin(q(t) + \hat{x}_3(t))/\cos(q(t)) \\ \hat{u}_1(t)\tan(\hat{u}_2(t)) \end{bmatrix}, \hat{y}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \hat{x}(t),$$

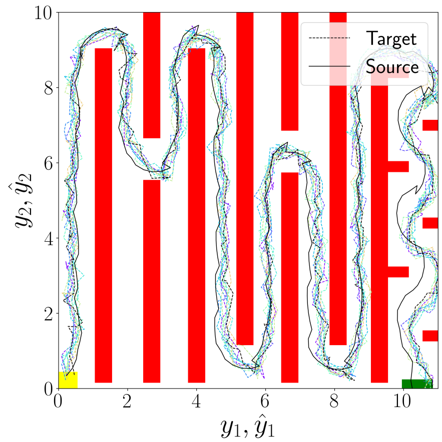
where $\hat{x} := [\hat{x}_1, \hat{x}_2, \hat{x}_3]$ is the state vector, in which $\hat{x}_1, \hat{x}_2, \hat{x}_3$ are horizontal position, vertical position, and steering angle, respectively, and $q(t) := \tan^{-1}(\tan \hat{u}_2(t)/2)$. Furthermore, $u_1, u_2 \in [-1, 1]$ are inputs to the system.

We consider $\hat{\mathcal{X}} = [0, 10] \times [0, 10] \times [-\pi, \pi]$, $\hat{\mathcal{X}}_0 = [0, 1] \times [0, 1] \times [0, 0.2]$, $\hat{U} = [-0.9, 0.9]^2$, which represent the state, initial state and input set of the source system, respectively. Moreover, $\mathcal{X} = \hat{\mathcal{X}} \times [-1, 1]^2$, $\mathcal{X}_0 = \hat{\mathcal{X}}_0 \times [-1, 1]^2$, $U = [-1, 1]^2$, represent the state, initial state and input set of the target system.

The corresponding Lipschitz constants are $\mathcal{L}_x = 1.1, \mathcal{L}_u = 0.1, \mathcal{L}_h = 1, \mathcal{L}_{\hat{x}} = 1.1, \mathcal{L}_{\hat{u}} = 0.1$, and $\mathcal{L}_{\hat{h}} = 1$. The training converged in 120 minutes with following parameters:



(a)



(b)

Figure 4: Vehicle control transfer from 3D to 5D. (a) The error between the outputs, and (b) the trajectories for both systems, for 10 different realizations.

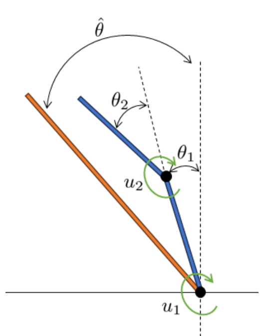
η=0.03, $\delta = 0.02$, $\mathfrak{e}$=0.002, $\mathcal{L}_V$=4, $N = \hat{N} = 200$, $M = 0.1$, $\alpha(x) = \log(x+1)$, and $\mathcal{L}_K$=4.1. In this case study, for $\alpha(\epsilon) = 1$, with 99% confidence, we get:

$$\mathbb{P}\big[\max_{t\in\mathbb{N}} \|h(x(t)) - \hat{h}(\hat{x}(t))\| \leq 1 | x_0, \hat{x}_0\big] \geq 0.9287,$$

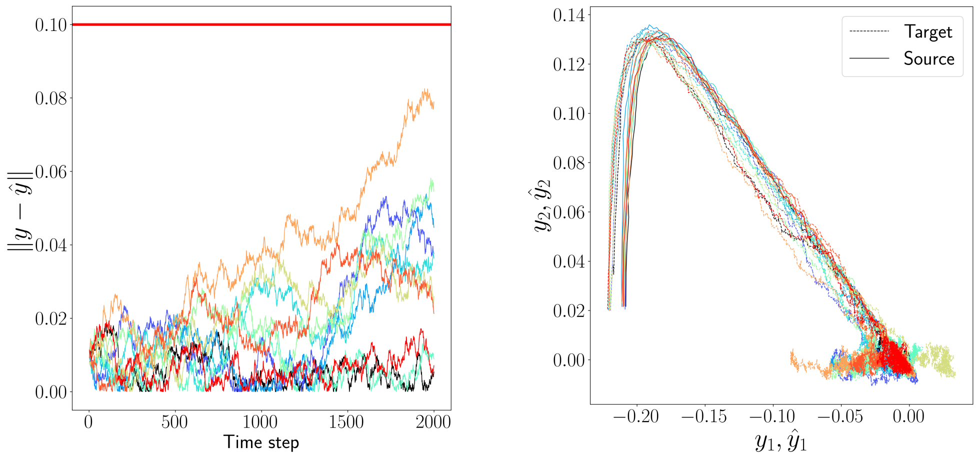
for $\|h(x(0)) - \hat{h}(\hat{x}(0))\| \leq 0.1$.

The error between outputs of source and target systems over an state sequence of 300 steps is depicted in Figure 4, for 10 different realizations. We leveraged the tool `SCOTS` (Rungger and Zamani, 2016) to design a controller for the source system, ensuring it reaches the goal (depicted by the green rectangle) while avoiding obstacles (depicted by red rectangles) from the initial set of states (depicted by the yellow rectangle). Note that applying `SCOTS` to the target system is infeasible due to its high dimensionality. We conducted these experiments with 10000 different realizations, and in only 52 cases did the difference between the outputs exceed 1, which aligns with the theoretical results.

B.2. Pendulum Control: From Single-Jointed to Double-Jointed



For our third case study, we transfer control from a single-jointed inverted pendulum to a double-jointed one as shown in the inset. This system serves as a classic benchmark in control theory due to its combination of inherent instability and nonlinearity, making it an ideal platform for assessing control transfer methods. Practical applications of the double inverted pendulum include bipedal locomotion in robotics, self-balancing vehicles, and crane load stabilization—all of which demand precise control of unstable, high-dimensional systems. The target system has the following model:



(a) (b)

Figure 5: Single-jointed pendulum control transfer to double-jointed pendulum. (a) The error between the outputs, and (b) the trajectories for both systems, for 10 different realizations.

$$\begin{bmatrix}\theta_1(t{+}1)\\ \omega_1(t{+}1)\\ \theta_2(t{+}1)\\ \omega_2(t{+}1)\end{bmatrix} = \begin{bmatrix}\theta_1(t){+}\tau\omega_1(t)\\ \omega_1(t){+}\tau(g\sin(\theta_1(t)){-}\sin(\theta_1(t){-}\theta_2(t))\omega_1^2(t))\\ \theta_2(t){+}\tau\omega_2(t)\\ \omega_2(t){+}\tau(g\sin(\theta_2(t)){+}\sin(\theta_1(t){-}\theta_2(t))\omega_2^2(t))\end{bmatrix} {+}\tau\begin{bmatrix}0 & 0\\ 30 & 0\\ 0 & 0\\ 0 & 39\end{bmatrix}U(t),$$

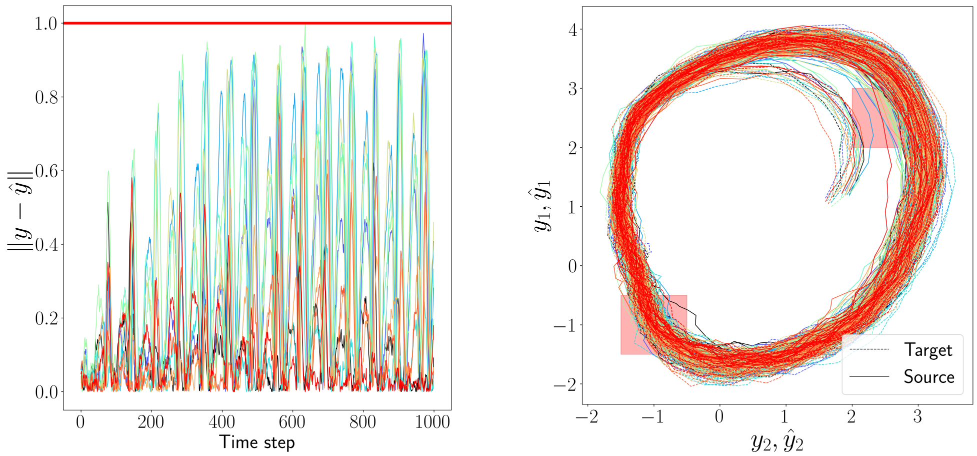
where $[\theta_1(t);\omega_1(t);\theta_2(t);\omega_2(t)] \in [-0.5, 0.5]^4$, and $y(t) = [\theta_1(t), \omega_1(t)]$ is the output. Here, θ_1 and θ_2 represent the angular position of the first and the second joint, respectively, and ω_1 and ω_2 are the angular velocity, respectively, and $U{\in}[-1, 1]^2$ are the inputs applied to the first and second joint, respectively. The initial set of states are $\mathcal{X}_0 = \mathcal{X}, \hat{\mathcal{X}}_0 = \hat{\mathcal{X}}$ for the target and the source systems, respectively. This is a simplified version of double inverted pendulum, where we assumed the second derivative of both angles are zero, to be able to discretize this system. The source system is an inverted pendulum with the following model:

$$\begin{bmatrix}\hat{\theta}(t+1)\\ \hat{\omega}(t+1)\end{bmatrix} = \begin{bmatrix}\hat{\theta}(t) + \tau\hat{\omega}(t)\\ \hat{\omega}(t) + \tau g\sin(\hat{\theta}(t))\end{bmatrix} + \tau\begin{bmatrix}0\\ 9.1\end{bmatrix}\hat{u}(t), \quad \hat{y}(t) = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\hat{x}(t),$$

where $[\hat{\theta}(t);\hat{\omega}(t)]{\in}[-0.5, 0.5]^2$ represent the angular position and velocity, respectively, and $\tau{=}0.01$ is the sampling time, and $\hat{U} = [-1, 1]$ is the input set. Furthermore, for both systems, $g = 9.8$ is the gravitational acceleration. The Lipschitz constants are $\mathcal{L}_x{=}1.098, \mathcal{L}_u = 0.39, \mathcal{L}_h{=}1, \mathcal{L}_{\hat{x}}{=}1.098, \mathcal{L}_{\hat{u}}{=}0.091,$ and $\mathcal{L}_{\hat{h}} = 1$.

The training converged in 150 minutes with following parameters: $\eta{=}0.01, \delta = 0.01, \hat{\mathfrak{e}} = 0.1, \mathfrak{e} = 0.001, \mathcal{L}_V = 1.2, N = \hat{N} = 100, M = 0.005, \alpha(x) = \log(x+1),$ and $\mathcal{L}_K{=}5.9$. In this case study, for $\alpha(\epsilon) = 0.1$, with 99% confidence, we get:

$$\mathbb{P}\big[\max_{t\in\mathbb{N}}\|h(x(t)) - \hat{h}(\hat{x}(t))\| \leq 0.1|x_0, \hat{x}_0\big] \geq 0.8567,$$



(a) (b)

Figure 6: Marine vessel control transfer from 3D to 6D. (a) The error between the outputs, and (b) the trajectories for both systems, for 10 different realizations.

for $\|h(x(0)) - \hat{h}(\hat{x}(0))\| \leq 0.01$.

The error between outputs of source and target systems over an state sequence of 2000 steps is depicted in Figure 3, for 10 different realizations. Source system is controlled by a simple proportional controller, and the setpoint was changed with every 1000 steps. We conducted these experiments with 10000 different realizations, and in only 867 cases did the difference between the outputs exceed 0.1, which aligns with the theoretical results. The source system is controlled by a formally correct neural control barrier certificate borrowed from (Nadali et al., 2025), which keeps the pendulum in the upright position.

B.3. Marine Vessel

Our final case study is the marine vessel system from (Meyer et al., 2020). The target system is a complex six-dimensional vessel, while the source system includes only its kinematic components. The target system has the following model:

$$\begin{aligned}
\eta(t+1) &= \eta(t) + \tau(R(\psi(t))\nu(t)),\\
\nu(t+1) &= \nu(t) + \tau M^{-1}\big(U(t) - C(\nu)\nu(t) - D\nu(t)\big),
\end{aligned}$$

where $\eta := [x, y, \psi]$ are the South-North and West-East positions and heading of the ship, and $\nu := [u; v; r]$ are the surge and sway velocities, and yaw rate of the ship. $R(\psi)$ is a rotation matrix, and $U \in \mathbb{R}^3$ is the control input affecting the three acceleration states of the ship. Moreover, M, D, C represent the inertia matrix including hydrodynamic added mass, damping matrix, and Coriolis matrix:

$$M = \begin{bmatrix} 87.4 & 0 & 0 \\ 0 & 98.3 & 2.48 \\ 0 & 2.48 & 22.2 \end{bmatrix}, C = u\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 98.3 \\ 0 & 0 & 2.48 \end{bmatrix}, D = \begin{bmatrix} 6.58 & 0 & 0 \\ 0 & 37.7 & 2.66 \\ 0 & 2.66 & 19.3 \end{bmatrix}.$$

The source system is only the kinematics part of the target system:

$$\hat{\eta}(t+1) = \hat{\eta}(t) + \tau R(\hat{\psi}(t))\hat{U}(t).$$

The output of both systems are South-North and West-East positions of both systems, respectively.

Due to the high dimensionality of the target and source systems, formal guarantees are infeasible as the sample complexity is prohibitively high. However, we present this case study to showcase the success of our training and provide empirical evidence of its correctness. Figure 6 illustrates the output sequences of both systems, over 10 realizations. We utilized the tool `SCOTS` to design a controller for the source system, ensuring infinite visits to both pink rectangles. Note that, applying `SCOTS` directly to the target system is infeasible due to its high dimensionality. This demonstrates the utility of our approach in enabling control transfer when traditional methods are impractical.