2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561348

# Efficient Reachability Analysis of Closed-Loop Systems with Neural Network Controllers

Michael Everett, Golnaz Habibi, Jonathan P. How

***Abstract*— Neural Networks (NNs) can provide major empirical performance improvements for robotic systems, but they also introduce challenges in formally analyzing those systems' safety properties. In particular, this work focuses on estimating the forward reachable set of closed-loop systems with NN controllers. Recent work provides bounds on these reachable sets, yet the computationally efficient approaches provide overly conservative bounds (thus cannot be used to verify useful properties), whereas tighter methods are too intensive for online computation. This work bridges the gap by formulating a convex optimization problem for reachability analysis for closed-loop systems with NN controllers. While the solutions are less tight than prior semidefinite program-based methods, they are substantially faster to compute, and some of the available computation time can be used to refine the bounds through input set partitioning, which more than overcomes the tightness gap. The proposed framework further considers systems with measurement and process noise, thus being applicable to realistic systems with uncertainty. Finally, numerical comparisons show that our approach based on linear programming and partitioning can give $10\times$ reduction in conservatism in $\frac{1}{2}$ of the computation time compared to the state-of-the-art, and the ability to handle various sources of uncertainty is highlighted on a quadrotor model.**

## I. Introduction

Neural Networks (NNs) are pervasive in robotics due to their ability to express highly general input-output relationships for perception, planning, and control tasks. However, before deploying NNs on safety-critical systems, there must be techniques to guarantee that the closed-loop behavior of systems with NNs will meet desired specifications. The goal of this paper is to develop a framework for guaranteeing that systems with NN controllers will reach their goal states while avoiding undesirable regions of the state space, as in Fig. 1.

Despite the importance of analyzing closed-loop behavior, much of the recent work on formal NN analysis has focused on NNs in isolation (e.g., for image classification) [1]–[6], with an emphasis on efficiently relaxing NN nonlinearities [7]–[13]. On the other hand, closed-loop system reachability has been studied for decades, but traditional approaches, such as Hamilton-Jacobi methods [14], [15], do not consider NNs in the loop.

A handful of recent works [16]–[21] propose methods that compute forward reachable sets of closed-loop systems with NN controllers. A key challenge is in maintaining computational efficiency while still providing tight bounds on the reachable sets. Moreover, the literature typically assumes perfect knowledge of system dynamics, with no stochasticity.

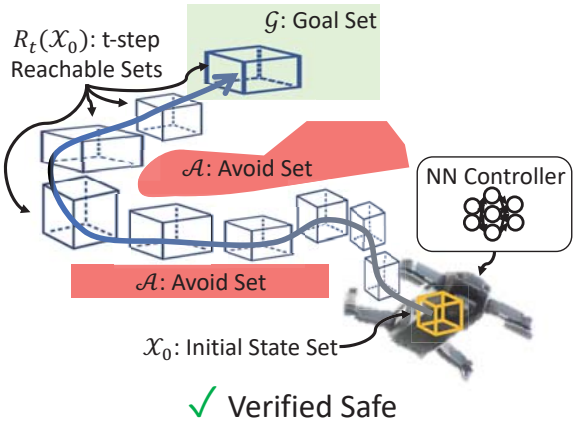



Fig. 1. Forward Reachability Analysis. The objective is to compute the blue sets $\mathcal{R}_t(\mathcal{X}_0)$, to ensure a system starting from $\mathcal{X}_0$ (yellow) ends in $\mathcal{G}$ (green) and avoids $\mathcal{A}_0, \mathcal{A}_1$ (red). This is especially challenging for systems with NN control policies.

To address the primary challenge of computational efficiency, we re-formulate the semi-definite program (SDP) from [21] as a linear program (LP) and leverage tools from [10]. While this relaxation provides substantial improvement in computational efficiency, it also introduces some conservatism. Thus, the proposed algorithm trades off some computational efficiency for bound tightness by partitioning the input set, as motivated by [22]. Finally, the proposed framework considers measurement and process noise throughout the formulation, thus being more amenable to applications on real, uncertain closed-loop systems.

This work's contributions include: (i) a convex optimization formulation of reachability analysis for closed-loop systems with NN controllers, providing a computationally efficient method for verifying safety properties, (ii) the use of input set partitioning techniques to provide tight bounds on the reachable sets despite large initial state sets, (iii) the consideration of measurement and process noise, which improves the applicability to real systems with uncertainty, and (iv) numerical comparisons with [21] showing $10\times$ tighter accuracy in $\frac{1}{2}$ the computation time via LP and partitioning, and an application on noisy quadrotor dynamics.

## II. Related Work

Related work on reachability analysis can be categorized into works on NNs in isolation, closed-loop systems without NNs, and closed-loop systems with NNs. For instance, machine learning literature includes many methods to verify

The authors are with Aerospace Controls Laboratory at Massachusetts Institute of Technology, {mfe, ghabibi, jhow}@mit.edu.
This work was supported by Ford Motor Company.
**Code:** https://github.com/mit-acl/nn_robustness_analysis

properties of NNs, often motivated by defending against adversarial examples [23]. These methods broadly range from exact [24] to tight [13] to efficient [10] to fast [7]. Although these tools are not designed for closed-loop systems, the NN relaxations from [10] provide a key foundation here.

For closed-loop systems, reachability analysis is a standard component of safety verification. Modern methods include Hamilton-Jacobi Reachability methods [14], [15], SpaceEx [25], Flow* [26], CORA [27], and C2E2 [28], [29], but these do not account for NN control policies. Orthogonal approaches that do not explicitly estimate the system's forward reachable set, but provide other notions of safety, include Lyapunov function search [30] and control barrier functions (CBFs) [31].

Recent reachability analysis approaches that do account for NN control policies face a tradeoff between computation time and conservatism. [16]–[18] use polynomial approximations of NNs to make the analysis tractable. Most works consider NNs with ReLU approximations, whereas [19] considers sigmoidal activations. [20], [32] introduce conservatism by assuming the NN controller could output its extreme values at every state. Most recently, [21] formulated the problem as a SDP, called Reach-SDP. This work builds on both [20], [21] and makes the latter more scalable by reformulating the SDP as a linear program, introduces sources of uncertainty in the closed-loop dynamics, and shows further improvements by partitioning the input set.

## III. Preliminaries

### A. Closed-Loop System Dynamics

Consider a discrete-time linear time-varying system,

$$\begin{aligned} \mathbf{x}_{t+1} &= A_t\mathbf{x}_t + B_t\mathbf{u}_t + \mathbf{c}_t + \boldsymbol{\omega}_t \\ \mathbf{y}_t &= C_t^T\mathbf{x}_t + \boldsymbol{\nu}_t, \end{aligned} \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^{n_x}$, $\mathbf{u}_t \in \mathbb{R}^{n_u}$, $\mathbf{y}_t \in \mathbb{R}^{n_y}$ are state, control, and output vectors, $A_t$, $B_t$, $C_t$ are known system matrices, $\mathbf{c}_t \in \mathbb{R}^{n_x}$ is a known exogenous input, and $\boldsymbol{\omega}_t \sim \Omega$ and $\boldsymbol{\nu}_t \sim N$ are process and measurement noises sampled at each timestep from unknown distributions with known, finite support (i.e., $\boldsymbol{\omega}_t \in [\underline{\boldsymbol{\omega}}_t, \bar{\boldsymbol{\omega}}_t], \boldsymbol{\nu}_t \in [\underline{\boldsymbol{\nu}}_t, \bar{\boldsymbol{\nu}}_t]$ element-wise).

We assume an output-feedback controller $\mathbf{u}_t = \pi(\mathbf{y}_t)$ parameterized by an $m$-layer feed-forward NN, optionally subject to control constraints, $\mathbf{u}_t \in \mathcal{U}_t$. We denote the closed-loop system with dynamics (1) and control policy $\pi$ as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t; \pi). \tag{2}$$

### B. Reachable Sets

For the closed-loop system (2), we denote $\mathcal{R}_t(\mathcal{X}_0)$ the forward reachable set at time $t$ from a given set of initial conditions $\mathcal{X}_0 \subseteq \mathbb{R}^{n_x}$, which is defined by the recursion

$$\mathcal{R}_{t+1}(\mathcal{X}_0) = f(\mathcal{R}_t(\mathcal{X}_0); \pi), \qquad \mathcal{R}_0(\mathcal{X}_0) = \mathcal{X}_0. \tag{3}$$

### C. Finite-Time Reach-Avoid Verification Problem

The finite-time reach-avoid properties verification is defined as follows: Given a goal set $\mathcal{G} \subseteq \mathbb{R}^{n_x}$, a sequence of avoid sets $\mathcal{A}_t \subseteq \mathbb{R}^{n_x}$, and a sequence of reachable set estimates $\mathcal{R}_t \subseteq \mathbb{R}^{n_x}$, determining that every state in the final estimated reachable set will be in the goal set and any state in the estimated reachable sets will not enter an avoid set requires computing set intersections, $\texttt{VERIFIED}(\mathcal{G}, \mathcal{A}_{0:N}, \mathcal{R}_{0:N}) \equiv \mathcal{R}_N \subseteq \mathcal{G}\ \&\ \mathcal{R}_t \cap \mathcal{A}_t = \emptyset, \forall t \in \{0, \dots, N\}$.

In the case of our nonlinear closed-loop system (2), where computing the reachable sets exactly is computationally intractable, we can instead compute outer-approximations of the reachable sets, $\bar{\mathcal{R}}(\mathcal{X}_0) \supseteq \mathcal{R}_t(\mathcal{X}_0)$. This is useful if the finite-time reach-avoid properties of the system as described by outer-approximations of the reachable sets are verified, because that implies the finite-time reach-avoid properties of the *exact* closed loop system are verified as well. Tight outer-approximations of the reachable sets are desirable, as they enable verification of tight goal and avoid set specifications, and they reduce the chances of verification being unsuccessful even if the exact system meets the specifications.

### D. Control Policy Neural Network Structure

Using notation from [10], for the $m$-layer neural network used in the control policy, the number of neurons in each layer is $n_k \forall k \in [m]$, where $[i]$ denotes the set $\{1, 2, \dots, i\}$. Let the $k$-th layer weight matrix be $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ and bias vector be $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$, and let $\Phi_k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_k}$ be the operator mapping from network input (measured output vector $\mathbf{y}_t$) to layer $k$. We have $\Phi_k(\mathbf{y}_t) = \sigma(\mathbf{W}^{(k)}\Phi_{k-1}(\mathbf{y}_t) + \mathbf{b}^{(k)}), \forall k \in [m-1]$, where $\sigma(\cdot)$ is the coordinate-wise activation function. The framework applies to general activations, including ReLU, $\sigma(\mathbf{z}) = \max(0, \mathbf{z})$. The network input $\Phi_0(\mathbf{y}_t) = \mathbf{y}_t$ produces the unclipped control input,

$$\mathbf{u}_t = \pi(\mathbf{y}_t) = \Phi_m(\mathbf{y}_t) = \mathbf{W}^{(m)}\Phi_{m-1}(\mathbf{y}_t) + \mathbf{b}^{(m)}. \tag{4}$$

### E. Neural Network Robustness Verification

A key step in quickly computing reachable sets of the closed-loop system (2) with a NN control policy is to relax nonlinear constraints induced by the NN's nonlinear activation functions. Within a known range of a neuron's input, a nonlinear activation can be linearly bounded above/below.

*Theorem 3.1 (From [10], Convex Relaxation of NN):* Given an $m$-layer neural network control policy $\pi : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$, there exist two explicit functions $\pi_j^L : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ and $\pi_j^U : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ such that $\forall j \in [n_m], \forall \mathbf{y} \in \mathcal{B}_p(\mathbf{y}_0, \epsilon)$, the inequality $\pi_j^L(\mathbf{y}) \le \pi_j(\mathbf{y}) \le \pi_j^U(\mathbf{y})$ holds true, where

$$\pi_j^U(\mathbf{y}) = \boldsymbol{\Lambda}_{j,:}^{(0)}\mathbf{y} + \sum_{k=1}^{m} \boldsymbol{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \boldsymbol{\Delta}_{:,j}^{(k)}) \tag{5}$$

$$\pi_j^L(\mathbf{y}) = \boldsymbol{\Omega}_{j,:}^{(0)}\mathbf{y} + \sum_{k=1}^{m} \boldsymbol{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \boldsymbol{\Theta}_{:,j}^{(k)}), \tag{6}$$

where $\mathbf{\Lambda}, \mathbf{\Omega}, \mathbf{\Delta}, \mathbf{\Theta}$ are defined recursively using NN weights and activations (e.g., ReLU, tanh), as detailed in [10].

In a closed-loop system, Theorem 3.1 bounds the control output for a *particular* measurement $\mathbf{y}$. Moreover, if all that is known is $\mathbf{y} \in \mathcal{B}_p(\mathbf{y}_0, \epsilon)$, Theorem 3.1 provides affine relationships between $\mathbf{y}$ and $\mathbf{u}$ (i.e., bounds valid within the known set of possible $\mathbf{y}$). These relationships enable efficient calculation of NN output bounds, using Corollary 3.3 of [10].

We could leverage [10] to compute reachable sets by first bounding the possible controls, then bounding the next state set by applying the extreme controls from each state. This is roughly the approach in [20], [32], for example. However, this introduces excessive conservatism, because both extremes of control would not be applied at every state (barring pathological examples). To produce tight bounds on the reachable sets, we leverage the relationship between measured output and control in Section IV.

## IV. Approach

Recall that our goal is to find the set of all possible next states, $\mathbf{x}_{t+1} \in \mathcal{X}_{\text{out}}$, given that the current state lies within a known set, $\mathbf{x}_t \in \mathcal{X}_{\text{in}}$. This will allow us to compute reachable sets recursively starting from an initial set $\mathcal{X}_{\text{in}} = \mathcal{X}_0$.

The approach follows the architecture in Fig. 2. After first relaxing the NN controller using Theorem 3.1, we then associate linearized extreme controllers with extreme next states in Section IV-B. Then, using the linearized extreme controller, we optimize over all states in the input set to find extreme next states in Section IV-C. We extend the formulation to handle control limits in Section IV-D, then describe how to convert the solutions of the optimization problems into reachable set descriptions in Section IV-E.

### A. Assumptions

This work assumes that $\mathcal{X}_{\text{in}}$ is described by either:

- an $\ell_p$ ball for some norm $p \in [1, \infty]$ and radius $\epsilon$, s.t. $\mathcal{X}_{\text{in}} = \mathcal{B}_p(\mathbf{x}_0, \epsilon)$; or
- a polytope, for some $\mathbf{A}^{\text{in}} \in \mathbb{R}^{m_{\text{in}} \times n_x}$, $\mathbf{b}^{\text{in}} \in \mathbb{R}^{m_{\text{in}}}$, s.t. $\mathcal{X}_{\text{in}} = \{\mathbf{x}_t | \mathbf{A}^{in} \mathbf{x}_t \leq \mathbf{b}^{in}\}$,

and shows how to compute $\mathcal{X}_{\text{out}}$ as described by either:

- an $\ell_\infty$ ball with radius $\epsilon$, s.t. $\mathcal{X}_{\text{out}} = \mathcal{B}_\infty(\mathbf{x}_0, \epsilon)$; or
- a polytope for a specified $\mathbf{A}^{\text{out}} \in \mathbb{R}^{m_{\text{out}} \times n_x}$, meaning we will compute $\mathbf{b}^{\text{out}} \in \mathbb{R}^{m_{out}}$ s.t. $\mathcal{X}_{\text{out}} = \{\mathbf{x}_t \in \mathbb{R}^{n_x} | \mathbf{A}^{\text{out}} \mathbf{x}_t \leq \mathbf{b}^{\text{out}}\}$.

We assume that either $\mathbf{A}^{\text{out}}$ is provided (in the case of polytope output bounds), or that $\mathbf{A}^{\text{out}} = \mathbf{I}_{n_x}$ (in the case of $\ell_\infty$ output bounds). Note that we use j to index the state vectors, j to index polytope facets, and $j$ to index the control vectors. Sections IV-B and IV-C assume that $\mathcal{U}_t = \mathbb{R}^{n_u}$ (no control input constraints) for cleaner notation; this assumption is relaxed in Section IV-D.

### B. Bounds on $\mathbf{x}_{t+1}$ from a particular $\mathbf{x}_t$

*Lemma 4.1:* Given an $m$-layer NN control policy $\pi : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$, closed-loop dynamics $f : \mathbb{R}^{n_x} \times \Pi \to \mathbb{R}^{n_x}$ as in Eqs. (1) and (2), and specification matrix $\mathbf{A}^{\text{out}} \in \mathbb{R}^{n_{\text{out}} \times n_x}$, for each j $\in [n_{\text{out}}]$, there exist two explicit functions $\pi_{:,\text{j}}^{L_{CL}} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ and $\pi_{:,\text{j}}^{U_{CL}} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ such that $\forall j \in [n_m], \forall \mathbf{x}_t \in \mathcal{B}_p(\mathbf{x}_{t,0}, \epsilon)$ and $\forall \mathbf{y}_t \in \mathcal{B}_\infty(C_t^T \mathbf{x}_t + \frac{\bar{\boldsymbol{\nu}}_t + \underline{\boldsymbol{\nu}}_t}{2}, \frac{\bar{\boldsymbol{\nu}}_t - \underline{\boldsymbol{\nu}}_t}{2})$, the inequality $\mathbf{A}_{\text{j},:}^{\text{out}} f(\mathbf{x}_t, \pi_{:,\text{j}}^{L_{CL}}) \leq \mathbf{A}_{\text{j},:}^{\text{out}} f(\mathbf{x}_t, \pi) \leq \mathbf{A}_{\text{j},:}^{\text{out}} f(\mathbf{x}_t, \pi_{:,\text{j}}^{U_{CL}})$ holds true, where

$$\pi_{:,\text{j}}^{U_{CL}}(\mathbf{y}_t) = \mathbf{\Upsilon}_{\text{j},:,:}^{(0)} \mathbf{y}_t + \mathbf{z}^U \tag{7}$$

$$\pi_{:,\text{j}}^{L_{CL}}(\mathbf{y}_t) = \mathbf{\Xi}_{\text{j},:,:}^{(0)} \mathbf{y}_t + \mathbf{z}^L, \tag{8}$$

letting

$$\mathbf{z}^U = \sum_{k=1}^{m} \left[ \mathbf{\Upsilon}_{\text{j},:,:}^{(k)} \mathbf{b}^{(k)} + \mathbb{1}_{n_u} \left( \left( \mathbf{\Upsilon}_{\text{j},:,:}^{(k)} \right)^T \odot \mathbf{\Psi}^{(k)} \right) \right] \tag{9}$$

$$\mathbf{z}^L = \sum_{k=1}^{m} \left[ \mathbf{\Xi}_{\text{j},:,:}^{(k)} \mathbf{b}^{(k)} + \mathbb{1}_{n_u} \left( \left( \mathbf{\Xi}_{\text{j},:,:}^{(k)} \right)^T \odot \mathbf{\Gamma}^{(k)} \right) \right] \tag{10}$$

and $\forall k \in [m]$, $\mathbf{\Upsilon}^{(k)} \in \mathbb{R}^{m_{\text{out}} \times n_u \times n_u}$, $\mathbf{\Psi}^{(k)} \in \mathbb{R}^{m_{\text{out}} \times n_k \times n_u}$,

$$\mathbf{\Upsilon}_{\text{j},:,:}^{(k)} = \bar{\mathbf{J}}_{\text{j},:,:}^{(k)} \mathbf{\Lambda}^{(k)} + \underline{\mathbf{J}}_{\text{j},:,:}^{(k)} \mathbf{\Omega}^{(k)} \tag{11}$$

$$\mathbf{\Psi}_{\text{j},:,:}^{(k)} = \mathbf{\Delta}^{(k)} \bar{\mathbf{J}}_{\text{j},:,:}^{(k)} + \mathbf{\Theta}^{(k)} \underline{\mathbf{J}}_{\text{j},:,:}^{(k)} \tag{12}$$

$$\mathbf{\Xi}_{\text{j},:,:}^{(k)} = \bar{\mathbf{J}}_{\text{j},:,:}^{(k)} \mathbf{\Omega}^{(k)} + \underline{\mathbf{J}}_{\text{j},:,:}^{(k)} \mathbf{\Lambda}^{(k)} \tag{13}$$

$$\mathbf{\Gamma}_{\text{j},:,:}^{(k)} = \mathbf{\Theta}^{(k)} \bar{\mathbf{J}}_{\text{j},:,:}^{(k)} + \mathbf{\Delta}^{(k)} \underline{\mathbf{J}}_{\text{j},:,:}^{(k)}. \tag{14}$$

using selector matrices $\bar{\mathbf{J}}^{(k)}, \underline{\mathbf{J}}^{(k)} \in \{0,1\}^{n_u \times n_k \times n_u}$,

$$\bar{\mathbf{J}}_{\text{j},j,:}^{(k)} = \begin{cases} \mathbf{e}_j^T, & \text{if } \mathbf{A}_{\text{j},:}^{\text{out}} B_{t,:,\text{j}} \geq 0 \\ \mathbf{0}^T, & \text{otherwise} \end{cases} \tag{15}$$

$$\underline{\mathbf{J}}_{\text{j},j,:}^{(k)} = \begin{cases} \mathbf{0}^T, & \text{if } \mathbf{A}_{\text{j},:}^{\text{out}} B_{t,:,\text{j}} \geq 0 \\ \mathbf{e}_j^T, & \text{otherwise} \end{cases}, \tag{16}$$

and $\mathbf{\Lambda}, \mathbf{\Omega}, \mathbf{\Delta}, \mathbf{\Theta}$ are computed from Theorem 3.1 with $\mathbf{y}_0 = C_t^T(\mathbf{x}_{t,0} + \frac{\bar{\boldsymbol{\nu}}_t + \underline{\boldsymbol{\nu}}_t}{2})$, and $\epsilon = \epsilon + \frac{\bar{\boldsymbol{\nu}}_t - \underline{\boldsymbol{\nu}}_t}{2}$.

*Proof:* For any particular measurement $\mathbf{y}_t$, after relaxing the NN according to Theorem 3.1, let $\Pi(\mathbf{y}_t) = \{\pi | \pi_j^L(\mathbf{y}_t) \leq \pi_j(\mathbf{y}_t) \leq \pi_j^U(\mathbf{y}_t) \forall j \in [n_u]\}$ denote the set of possible effective control policies. Denote the control policy $\pi_{:,\text{j}}^{U_{CL}} \in \Pi(\mathbf{y}_t)$ as one that induces the least upper bound on the j-th facet of the next state polytope,

$$\begin{aligned} &\mathbf{A}_{\text{j},:}^{\text{out}} f(\mathbf{x}_t; \pi_{:,\text{j}}^{U_{CL}}) = \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}_{\text{j},:}^{\text{out}} f(\mathbf{x}_t; \pi) \\ &= \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}_{\text{j},:}^{\text{out}} \left[ A_t \mathbf{x}_t + B_t \pi(\mathbf{y}_t) + \mathbf{c}_t + \boldsymbol{\omega}_t \right] \\ &= \left[ \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}_{\text{j},:}^{\text{out}} B_t \pi(\mathbf{y}_t) \right] + \mathbf{A}_{\text{j},:}^{\text{out}} \left[ A_t \mathbf{x}_t + \mathbf{c}_t + \boldsymbol{\omega}_t \right], \end{aligned} \tag{17}$$

Thus for $\mathbf{y}_t$,

$$\pi_{:,\text{j}}^{U_{CL}} = \underset{\pi \in \Pi(\mathbf{y}_t)}{\operatorname{argmax}} \mathbf{A}_{\text{j},:}^{\text{out}} B_t \pi(\mathbf{y}_t). \tag{18}$$

The resulting control input $\forall$j $\in [m_{t+1}], j \in [n_u]$ is,

$$\pi_{j,\text{j}}^{U_{CL}}(\mathbf{y}_t) = \begin{cases} \pi_j^U(\mathbf{y}_t), & \text{if } \mathbf{A}_{\text{j},:}^{\text{out}} B_{t,:,j} \geq 0 \\ \pi_j^L(\mathbf{y}_t), & \text{otherwise} \end{cases}. \tag{19}$$

Writing (19) in matrix form results in (7). The proof of the lower bound follows similarly. ■

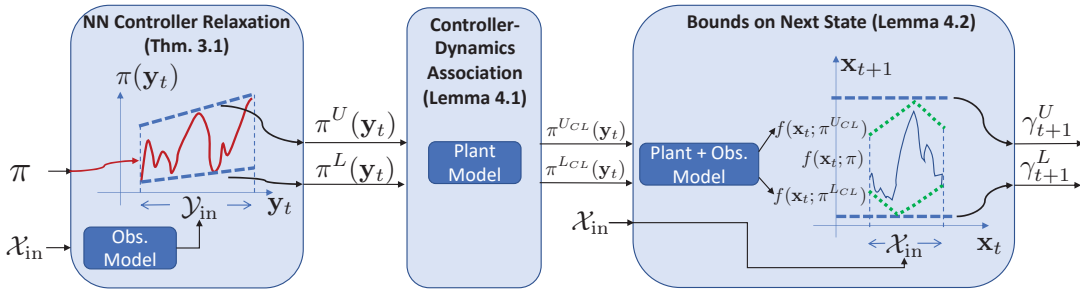



Fig. 2. Approach Overview for simple 1D system. Theorem 3.1 relaxes the NN to give affine relationships between observation $\mathbf{y}_t$ and control: $\pi^U, \pi^L$. Lemma 4.1 uses the system dynamics to associate $\pi^U, \pi^L$ with the next state set. Lemma 4.2 optimizes the closed-loop dynamics over all states $\mathbf{x}_t \in \mathcal{X}_{\text{in}}$ to compute bounds on the next state, $\gamma^U_{t+1}, \gamma^L_{t+1}$.

### C. Bounds on $\mathbf{x}_{t+1}$ from any $\mathbf{x}_t \in \mathcal{X}_{in}$

Now that we can bound each facet of the next state polytope given a particular current state and observation, we can form bounds on the next state polytope facet given a *set* of possible current states. This is necessary to handle initial state set constraints and to compute "$t > 1$"-step reachable sets recursively as in (3). We assume $\mathbf{x}_t \in \mathcal{X}_{\text{in}}$.

*Lemma 4.2:* Given an $m$-layer NN control policy $\pi : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$, closed-loop dynamics $f : \mathbb{R}^{n_x} \times \Pi \to \mathbb{R}^{n_x}$ as in Eqs. (1) and (2), and specification matrix $\mathbf{A}^{\text{out}} \in \mathbb{R}^{n_{\text{out}} \times n_x}$, for each $\text{j} \in [n_{\text{out}}]$, there exist two fixed values $\gamma^U_{t+1,\text{j}}$ and $\gamma^L_{t+1,\text{j}}$ such that $\forall \mathbf{x}_t \in \mathcal{X}_{\text{in}}$, the inequality $\gamma^L_{t+1,\text{j}} \leq \mathbf{A}^{\text{out}}_{\text{j},:} f(\mathbf{x}_t; \pi) \leq \gamma^U_{t+1,\text{j}}$ holds true, where

$$\gamma^U_{t+1,\text{j}} = \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{M}^U_{\text{j},:} \mathbf{x}_t + \mathbf{n}^U_{\text{j}} \tag{20}$$

$$\gamma^L_{t+1,\text{j}} = \min_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{M}^L_{\text{j},:} \mathbf{x}_t + \mathbf{n}^L_{\text{j}}, \tag{21}$$

with $\mathbf{M}^U \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{n}^U \in \mathbb{R}^{n_x}$ defined as

$$\mathbf{M}^U_{\text{j},:} = \left(\mathbf{A}^{\text{out}}_{\text{j},:} \left(A_t + B_t \boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} C_t^T\right)\right) \tag{22}$$

$$\mathbf{M}^L_{\text{j},:} = \left(\mathbf{A}^{\text{out}}_{\text{j},:} \left(A_t + B_t \boldsymbol{\Psi}^{(0)}_{\text{j},:,:} C_t^T\right)\right) \tag{23}$$

$$\mathbf{n}^U_{\text{j}} = \mathbf{A}^{\text{out}}_{\text{j},:} \left(B_t \left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \left(\bar{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\nu}}_t + \underline{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \bar{\mathbf{J}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\omega}}_t + \underline{\mathbf{J}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\omega}}_t\right) \tag{24}$$

$$\mathbf{n}^L_{\text{j}} = \mathbf{A}^{\text{out}}_{\text{j},:} \left(B_t \left(\boldsymbol{\Psi}^{(0)}_{\text{j},:,:} \left(\underline{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\nu}}_t + \bar{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \underline{\mathbf{J}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\omega}}_t + \bar{\mathbf{J}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\omega}}_t\right), \tag{25}$$

and where $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$, $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$ are defined as in Eqs. (15) and (16), but using $\mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,\text{j}} \boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} C_t^T$ and $\mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,\text{j}} \boldsymbol{\Psi}^{(0)}_{\text{j},:,:} C_t^T$, respectively, with $\boldsymbol{\Upsilon}, \boldsymbol{\Psi}, \mathbf{z}^U, \mathbf{z}^L, \bar{\mathbf{J}}, \underline{\mathbf{J}}$ computed from Lemma 4.1.

*Proof:* Bound the next state polytope's j-th facet above,

$$\mathbf{A}^{\text{out}}_{\text{j},:} \mathbf{x}_{t+1} = \mathbf{A}^{\text{out}}_{\text{j},:} f(\mathbf{x}_t; \pi) \tag{26}$$

$$\leq \mathbf{A}^{\text{out}}_{\text{j},:} f(\mathbf{x}_t; \pi^{U_{CL}}_{:,\text{j}}) \tag{27}$$

$$\leq \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\text{j},:} f(\mathbf{x}_t; \pi^{U_{CL}}_{:,\text{j}}) := \gamma^U_{t+1,\text{j}} \tag{28}$$

$$= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\text{j},:} \left[A_t \mathbf{x}_t + B_t \pi^{U_{CL}}_{:,\text{j}}(\mathbf{y}_t) + c_t + \boldsymbol{\omega}_t\right] \tag{29}$$

$$= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\text{j},:} \left[A_t \mathbf{x}_t + B_t \left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \mathbf{y}_t + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right] \tag{30}$$

$$= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\text{j},:} \left[A_t \mathbf{x}_t + B_t \left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \left(C_t^T \mathbf{x}_t + \boldsymbol{\nu}_t\right) + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right] \tag{31}$$

$$= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \left(\mathbf{A}^{\text{out}}_{\text{j},:} \left(A_t + B_t \boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} C_t^T\right)\right) \mathbf{x}_t + \mathbf{A}^{\text{out}}_{\text{j},:} \left(B_t \left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \boldsymbol{\nu}_t + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right) \tag{32}$$

$$= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \left(\mathbf{A}^{\text{out}}_{\text{j},:} \left(A_t + B_t \boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} C_t^T\right)\right) \mathbf{x}_t + \mathbf{A}^{\text{out}}_{\text{j},:} \left(B_t \left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \left(\bar{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\nu}}_t + \underline{\bar{\mathbf{J}}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + c_t + \bar{\mathbf{J}}^{(0)}_{\text{j},:,:} \bar{\boldsymbol{\omega}}_t + \underline{\mathbf{J}}^{(0)}_{\text{j},:,:} \underline{\boldsymbol{\omega}}_t\right), \tag{33}$$

where (30) substitutes the definition of $\pi^{U_{CL}}_{:,\text{j}}$ from Lemma 4.1, (31) substitutes the observation from (2), (32) separates terms that depend on $\mathbf{x}_t$, and (33) introduces the worst-case realizations of process and measurement noise. Substituting $\mathbf{M}^U_{\text{j},:}, \mathbf{n}^U_{\text{j}}$ results in (20). The proof of the lower bound follows similarly. ■

The optimization problems in Eqs. (20) and (21) have convex cost with convex constraints $\mathbf{x}_t \in \mathcal{X}_{\text{in}}$ (e.g., polytope $\mathcal{X}_{in}$). We solve the linear programs (LPs) with `cvxpy` [33],

$$\gamma^U_{t+1,\text{j}} = \texttt{LP}(\mathbf{M}^U_{\text{j},:} \mathbf{x}_t, \mathbf{A}^{\text{in}}, \mathbf{b}^{\text{in}}) + \mathbf{n}^U_{\text{j}} \tag{34}$$

$$\gamma^L_{t+1,\text{j}} = \texttt{LP}(-\mathbf{M}^L_{\text{j},:} \mathbf{x}_t, \mathbf{A}^{\text{in}}, \mathbf{b}^{\text{in}}) + \mathbf{n}^L_{\text{j}}. \tag{35}$$

### D. Accounting for Control Limits, $\mathcal{U}_t$

The key terms in Lemma 4.1 can be modified to account for control input constraints, as

$$\pi^{U_{CL}}_{:,\text{j}}(\mathbf{y}_t) = \text{Proj}_{\mathcal{U}_t}\left(\boldsymbol{\Upsilon}^{(0)}_{\text{j},:,:} \mathbf{y}_t + \mathbf{z}^U\right) \tag{36}$$

$$\pi^{L_{CL}}_{:,\text{j}}(\mathbf{y}_t) = \text{Proj}_{\mathcal{U}_t}\left(\boldsymbol{\Xi}^{(0)}_{\text{j},:,:} \mathbf{y}_t + \mathbf{z}^L\right), \tag{37}$$

A common example is box control input constraints. The element-wise control input is,

$$\pi^{U_{CL}}_{j,\text{j}}(\mathbf{y}_t) = \begin{cases} \text{clip}(\pi^U_j(\mathbf{y}_t), \underline{\mathbf{u}}_j, \bar{\mathbf{u}}_j), & \text{if } \mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,j} \geq 0 \\ \text{clip}(\pi^L_j(\mathbf{y}_t), \underline{\mathbf{u}}_j, \bar{\mathbf{u}}_j), & \text{otherwise} \end{cases}, \tag{38}$$

where clip saturates the control if it exceeds the limits. However, this could be non-convex depending on the domain of $\mathbf{x}_t$ (and violates DCP rules in `cvxpy` [33] regardless). In this work, we only apply part of the control input constraint,

$$\pi^{U_{CL}}_{j,\text{j}}(\mathbf{y}_t) = \begin{cases} \min(\pi^U_j(\mathbf{y}_t), \bar{\mathbf{u}}_j), & \text{if } \mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,j} \geq 0 \\ \max(\pi^L_j(\mathbf{y}_t), \underline{\mathbf{u}}_j), & \text{otherwise} \end{cases}, \tag{39}$$

and raise an error if the other limit is violated (which did not happen in our experiments). Future work will investigate solutions via convex relaxations [34] of the clip function.

### E. Converting State Constraints into Reachable Sets

*1) Reachable Sets as $\ell_\infty$ balls:* Assume $\mathcal{X}_0$ is an $\ell_p$ ball. Define $\{p, \epsilon, \mathbf{x}_0\}$ s.t. $\mathcal{X}_0 \subseteq \mathcal{B}_p(\mathbf{x}_0, \epsilon)$ and let $\bar{\mathcal{R}}_0(\mathcal{X}_0) = \mathcal{X}_0$. Using the results of the previous section, use $\mathbf{x}_{t=0} \in \mathcal{B}_p(\mathbf{x}_0, \epsilon)$ to compute $(\gamma^L_{1,\mathrm{j}}, \gamma^U_{1,\mathrm{j}})$ for each index of the state vector $\mathrm{j} \in [n_x]$, specifying $\mathbf{A}^{\text{out}} = \mathbf{I}_{n_x}$. Recursively compute

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \mathcal{B}_\infty \left( \frac{\gamma^U_{t+1,:} + \gamma^L_{t+1,:}}{2}, \frac{\gamma^U_{t+1,:} - \gamma^L_{t+1,:}}{2} \right). \quad (40)$$

*2) Reachable Sets as Polytopes:* Assume $\mathcal{X}_0$ is an $\ell_p$ ball or polytope. Either define $\{p, \epsilon, \mathbf{x}_0\}$ s.t. $\mathcal{X}_0 \subseteq \mathcal{B}_p(\mathbf{x}_0, \epsilon)$ or define $\{\mathbf{A}^{\text{in}}, \mathbf{b}^{\text{in}}\}$ s.t. $\mathcal{X}_{\text{in}} = \{\mathbf{x}_t | \mathbf{A}^{in}\mathbf{x}_t \leq \mathbf{b}^{in}\}$ and let $\bar{\mathcal{R}}_0(\mathcal{X}_0) = \mathcal{X}_0$. Using the results of the previous section, use $\mathbf{x}_{t=0} \in \mathcal{B}_p(\mathbf{x}_0, \epsilon)$ or $\{\mathbf{A}^{\text{in}}, \mathbf{b}^{\text{in}}\}$ to compute $(\gamma^L_{1,\mathrm{j}}, \gamma^U_{1,\mathrm{j}})$ for each index of output polytope facets $\mathrm{j} \in [m_{out}]$, giving

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \{\mathbf{x}_t | \begin{bmatrix} \mathbf{A}^{\text{out}} \\ -\mathbf{A}^{\text{out}} \end{bmatrix} \mathbf{x}_t \leq \begin{bmatrix} \gamma^U_{t+1,:} \\ -\gamma^L_{t+1,:} \end{bmatrix} \}. \quad (41)$$

In both cases, $\bar{\mathcal{R}}_t(\mathcal{X}_0) \supseteq \mathcal{R}_t(\mathcal{X}_0) \forall t \geq 0$, so these $\bar{\mathcal{R}}_t$ can be used to verify the original closed loop system (2).

### F. Tighter Reachable Sets via Partitioning the Input Set

NN relaxation methods can be improved by partitioning the input set, particularly when the input set is large and of low dimension. Here, we achieve tighter bounds by splitting $\mathcal{X}_0$ into several subsets, computing $N$-step reachable sets for each of the subsets separately, then returning the union of all reachable sets from each subset. This idea falls into the general framework from [35] of choosing a propagator and a partitioner (e.g., uniform [36]) for the analysis, where Reach-LP/SDP represent propagators for closed-loop systems.

## V. Numerical Experiments

This section demonstrates our convex reachability analysis tool, Reach-LP, on simulated scenarios. We first show an example verification task and quantify the improvement in runtime vs. bound tightness over the state-of-the-art [21] for a double integrator system. We then apply the algorithm on a 6D quadrotor model subject to multiple sources of noise.

### A. Double Integrator

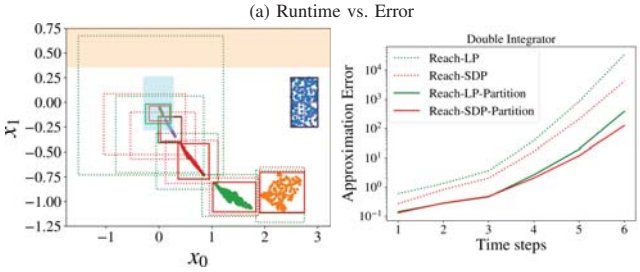
Consider the LTI double integrator system from [21],

$$\mathbf{x}_{t+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A_t} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0.5 \\ 1 \end{bmatrix}}_{B_t} \mathbf{u}_t, \quad (42)$$

with $\mathbf{c}_t = 0$, $C_t = I_2$ and no noise, discretized with sampling time $t_s = 1s$. As in [21], we implemented a linear MPC with prediction horizon $N_{MPC} = 10$, weighting matrices $Q = I_2, R = 1$, and terminal weighting matrix $P_\infty$ synthesized from the discrete-time Riccati equation, subject to state constraints $\mathcal{A}^C = [-5, 5] \times [-1, 1]$ and input constraint $\mathbf{u}_t \in [-1, 1] \forall t$. We used MPC to generate 2420 samples of state and input pairs then trained a NN with Keras [37] for 20 epochs with batch size 32.

| Algorithm | Runtime [s] | Error |
|---|---|---|
| Reach-SDP [21] | 20.31 | 206 |
| Reach-SDP-Partition | 347.14 | **19.35** |
| Reach-LP | **0.63** | 848 |
| Reach-LP-Partition | 9.87 | 19.87 |

(a) Runtime vs. Error




(b) Reachable Set Estimates (c) Over-approximation error

Fig. 3. Reachable Sets for Double Integrator. In (a), Reach-LP is 30× faster to compute but 4× looser than Reach-SDP [21]. Reach-LP-Partition refines the Reach-LP bounds by splitting the input set into 16 subsets, giving 10× faster computation time and 2× tighter bounds than Reach-SDP [21]. In (b), all reachable set algorithms bound sampled states across the timesteps, starting from the blue $\mathcal{X}_0$, and the tightness of these bounds is quantified per timestep in (c).

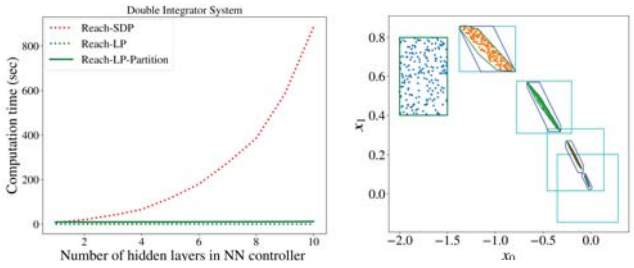
### B. Comparison with Baseline

Fig. 3 compares several algorithms on the double integrator system using a NN with [5,5] neurons and ReLU activations. The key takeaway is that Reach-LP-Partition provides a 10× improvement in reachable set tightness over the prior state-of-the-art, Reach-SDP [21] (which does not use input set partitioning), while requiring $\frac{1}{2}$ of the computation time. We implemented Reach-SDP in Python with `cvxpy` and `MOSEK` [38]. All computation times are reported from an i7-6700HQ CPU with 16GB RAM.

Fig. 3b shows sampled trajectories, where each colored cluster of points represents sampled reachable states at a particular timestep (blue→orange→green, etc.). Recall that sampling trajectories could miss possible reachable states, whereas these algorithms are guaranteed to over-approximate the reachable sets. Reachable set bounds are visualized for various algorithms: Reach-SDP [21], Reach-LP, and those two algorithms after partitioning the input set into 16 cells. The key takeaway is that while all approaches provide outer bounds on the sampled trajectories, the algorithms provide various degrees of tightness to the sampled points.

We quantify tightness as the ratio of areas between the smallest axis-aligned bounding box on the sampled points and the provided reachable set (minus 1), shown in Fig. 3c as the system progresses forward in time. Note that as expected, all algorithms get worse as the number of timesteps increase, but that Reach-LP-Partition and Reach-SDP-Partition perform the best and similarly. This provides numerical comparisons of the rectangle sizes from Fig. 3b.

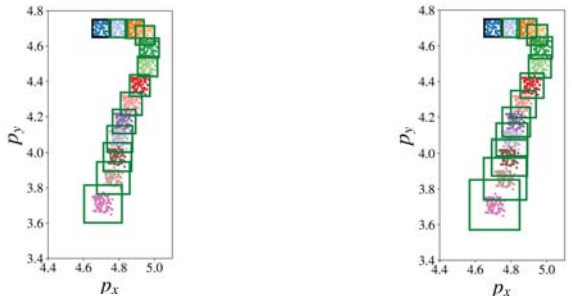
Note that both Reach-LP and Reach-SDP methods could be improved by properly choosing the direction of polytope facets. Additionally, while Reach-SDP can provide ellipsoidal bounds given the quadratic nature of the formulation, we implement only the polytope bounds in this comparison.




(a) Runtime with NN depth

(b) Number of Polytope Facets

Fig. 4. (a) Our linear relaxation-based methods (Reach-LP, Reach-LP-Partition) scale well for deeper NNs (Reach-LP: 0.6 to 0.74s), whereas SDP-based methods grow to intractable runtimes. Note that input set partitioning multiplies computation time by a scalar.(b) Using Reach-LP, the bounding shapes correspond to $l_\infty$-ball, 8-Polytope, and 35-Polytope. Reachable sets become tighter with more facets.



(a) No Noise

(b) Sensor & Process Noise

Fig. 5. Reachable Sets for 6D Quadrotor. Only $(x, y)$ states are shown, even though the reachable sets are computed in 6D. Green boxes (Reach-LP) bound the clusters of sampled points at each discrete timestep, starting from the blue $\mathcal{X}_0$. It took 4.89 sec to compute the 12 reachable sets per scenario. In (b), $\boldsymbol{\nu} \sim \texttt{Unif}(\pm 0.001 \cdot \mathbb{1}_6), \boldsymbol{\omega} \sim \texttt{Unif}(\pm 0.005 \cdot \mathbb{1}_6)$.

### C. Verification

A primary application of reachable sets is to verify reach-avoid properties. In Fig. 3b, we consider a case with an avoid set $\mathcal{A} = \{\mathbf{x} | \mathbf{x}_1 \geq 0.35\}$ (orange) and a goal set $\mathcal{G} = [-0.25, 0.25] \times [-0.25, 0.25]$ (cyan). Each algorithm, except Reach-LP, verifies these properties for this 5-step scenario, highlighting the importance of tight reachable sets.

### D. Scalability to Deep NNs

To demonstrate the scalability of the method, we trained NNs with 1-10 hidden layers of 5 neurons and report the average runtime of 5 trials of reachability analysis of the double integrator system. In Fig. 4a, while Reach-SDP appears to grow exponentially (taking $> 800s$ for a 10-layer NN), our proposed Reach-LP methods remain very efficient ($< 0.75s$ for Reach-LP on all NNs). Note that we omit Reach-SDP-Partition ($\sim 16\times$ more than Reach-SDP) from this plot to maintain reasonable scale.

### E. Ablation Study: $\ell_\infty$ vs. Polytopes

Recall that Section IV-E described reachable sets as either polytopes or $\ell_\infty$-balls. Fig. 4b shows the effect of that choice: as the number of sides of the polytope increases, the reachable set size decreases. The tradeoff is that the computation time scales linearly with the number of sides on the polytope. Note that a $\ell_\infty$-ball is a 4-polytope, and that $\mathcal{X}_0$ was chosen to show a different scenario than Fig. 3.

### F. 6D Quadrotor with Noise

Consider the 6D nonlinear quadrotor from [21], [39],

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0_{3\times3} & I_3 \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}}_{A_t} \mathbf{x}_t + \underbrace{\begin{bmatrix} & g & 0 & 0 \\ 0_{3\times3} & 0 & -g & 0 \\ & 0 & 0 & 1 \end{bmatrix}^T}_{B_t} \underbrace{\begin{bmatrix} \tan(\theta) \\ \tan(\phi) \\ \tau \end{bmatrix}}_{\mathbf{u}_t} + \underbrace{\begin{bmatrix} 0_{5\times1} \\ -g \end{bmatrix}}_{\mathbf{c}_t} + \boldsymbol{\omega}_t, \tag{43}$$

which differs from [21], [39] in that we add $\boldsymbol{\omega}_t$ as a uniform process noise, and that the output is measured as in (1) with $C_t = I_6$, subject to uniform sensor noise. As in [21], the state vector contains 3D positions and velocities, $[p_x, p_y, p_z, v_x, v_y, v_z]$, while nonlinearities from [39] are absorbed into the control as functions of $\theta$ (pitch), $\phi$ (roll), and $\tau$ (thrust) (subject to the same actuator constraints as [21]). We implemented a similar nonlinear MPC as [21] in MATLAB to collect $(\mathbf{x}_t, \mathbf{u}_t)$ training pairs, then trained a [32,32] NN with Keras as above. We use Euler integration to account for (43) in our discrete time formulation.

Fig. 5 shows the reachable sets with and without noise. Note that while these plots only show $(x, y)$ position, the reachable sets are estimated in all 6D. The first key takeaway is that the green boxes (Reach-LP with $\ell_\infty$-balls) provide meaningful bounds for a long horizon (12 steps, $1.2s$ shown). Secondly, unlike Reach-SDP, Reach-LP is guaranteed to bound worst-case noise realizations.

## VI. Future Directions

Many open questions remain in analyzing closed-loop systems with NN controllers. How to mitigate the conservatism due to the accumulation of approximation error over many timesteps? Can similar methods be developed for nonlinear systems or systems with uncertainty in $A_t$ or $B_t$? Can the ideas be extended naturally to continuous time systems, rather than through Euler integration? How to handle the non-convex nature of saturations for control limits? What partitioning scheme is best for closed-loop reachability?

## VII. Conclusion

This paper proposed a convex relaxation-based algorithm for computing forward reachable sets of closed-loop systems with NN controllers. Prior work is limited to shallow NNs and is computationally intensive, which limits applicability to real systems. Furthermore, our method accounts for measurement of sensor and process noise as demonstrated on a quadrotor model. The results show that this work advances the state-of-the-art in guaranteeing properties of systems that employ NNs in the feedback loop.

## References

[1] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *ATVA*, 2017.

[2] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pp. 97–117, 2017.

[3] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification* (R. Majumdar and V. Kunčak, eds.), (Cham), pp. 3–29, Springer International Publishing, 2017.

[4] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *CoRR*, vol. abs/1706.07351, 2017.

[5] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations (ICLR)*, 2019.

[6] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, May 2018.

[7] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[8] T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning (ICML)*, 2018.

[9] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," in *Advances in Neural Information Processing Systems*, pp. 10802–10813, 2018.

[10] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in neural information processing systems*, pp. 4939–4948, 2018.

[11] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *ICML*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 5283–5292, 2018.

[12] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[13] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *arXiv preprint arXiv:1903.01287*, 2019.

[14] C. J. Tomlin, J. Lygeros, and S. S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[15] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253, IEEE, 2017.

[16] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 157–168, 2019.

[17] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "Reachnn: Reachability analysis of neural-network controlled systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.

[18] J. Fan, C. Huang, X. Chen, W. Li, and Q. Zhu, "Reachnn*: A tool for reachability analysis of neural-network controlled systems," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 537–542, Springer, 2020.

[19] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 169–178, 2019.

[20] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[21] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," in *59th IEEE Conference on Decision and Control*, 2020.

[22] W. Xiang, H.-D. Tran, J. A. Rosenfeld, and T. T. Johnson, "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *2018 Annual American Control Conference (ACC)*, pp. 1574–1579, IEEE, 2018.

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[24] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, pp. 97–117, Springer, 2017.

[25] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*, pp. 379–395, Springer, 2011.

[26] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification*, pp. 258–263, Springer, 2013.

[27] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[28] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2e2: A verification tool for stateflow models," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 68–82, Springer, 2015.

[29] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with c2e2," in *International Conference on Computer Aided Verification*, pp. 531–538, Springer, 2016.

[30] A. Papachristodoulou and S. Prajna, "On the construction of lyapunov functions using the sum of squares decomposition," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 3, pp. 3482–3487, IEEE, 2002.

[31] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[32] G. Yang, G. Qian, P. Lv, and H. Li, "Efficient verification of control systems with neural network controllers," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, pp. 1–7, 2019.

[33] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[34] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020.

[35] M. Everett, G. Habibi, and J. P. How, "Robustness analysis of neural networks via efficient partitioning with applications in control systems," *IEEE Control Systems Letters*, 2020.

[36] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5777–5783, 2018.

[37] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[38] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm," in *High performance optimization*, pp. 197–232, Springer, 2000.

[39] D. M. Lopez, P. Musau, H.-D. Tran, and T. T. Johnson, "Verification of closed-loop systems with neural network controllers," *EPiC Series in Computing*, vol. 61, pp. 201–210, 2019.