# REACHABILITY ANALYSIS OF NEURAL FEEDBACK LOOPS

**Michael Everett, Golnaz Habibi, Jonathan P. How**
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{mfe, ghabibi, jhow}@mit.edu

## ABSTRACT

Although neural networks (NNs) are traditionally trained to learn an input-output relationship from a dataset (e.g., classification, regression), real world deployments of machine learning (ML) are often part of complex, closed-loop systems, or *neural feedback loops*. For instance, robots with learned policies take actions that modify the world state and influence future observations. Thus, formal analysis of how NN decisions will play out in closed-loop systems is a fundamental component of robust and reliable ML in the real world. We address this issue by developing an algorithm which computes *forward reachable sets* – bounds on the set of states a system could reach in the future – using convex optimization. The proposed framework accounts for several sources of uncertainty that are present in the real world, such as in the initial conditions, and from measurement and process noise. Numerical experiments show $10\times$ reduction in conservatism (which enables verification of tighter safety properties) 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.

## 1 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, there must be techniques to guarantee that the closed-loop behavior of systems with NNs will meet desired specifications. This paper develops a framework that guarantees 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) (Ehlers, 2017; Katz et al., 2017b; Huang et al., 2017; Lomuscio & Maganti, 2017; Tjeng et al., 2019; Gehr et al., 2018), with an emphasis on efficiently relaxing NN nonlinearities (Gowal et al., 2018; Weng et al., 2018; Singh et al., 2018; Zhang et al., 2018; Wong & Kolter, 2018; Raghunathan et al., 2018; Fazlyab et al., 2019). On the other hand, closed-loop system reachability has been studied for decades, but traditional approaches, such as Hamilton-Jacobi methods (Tomlin et al., 2000; Bansal et al., 2017), do not consider NNs in the loop.

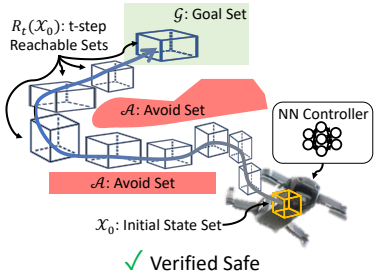



Figure 1. Reachability Analysis. The objective is to compute the blue sets $\mathcal{R}_t(\mathcal{X}_0)$, to ensure a system starting from $\mathcal{X}_0$ (yellow) ends in $\mathcal{G}$ (green) and avoids $\mathcal{A}_0, \mathcal{A}_1$ (red).

A handful of recent works (Dutta et al., 2019; Huang et al., 2019; Fan et al., 2020; Ivanov et al., 2019; Xiang et al., 2020; Hu et al., 2020) 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 that the system dynamics are known perfectly, with no stochasticity.

To address the primary challenge of computational efficiency, we re-formulate the semi-definite program (SDP) from (Hu et al., 2020) as a linear program and leverage tools from (Zhang et al., 2018). 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 (Xiang et al., 2018). Finally, the proposed framework considers measurement and process noise throughout the formulation, thus being more amenable to applications on real, uncertain closed-loop systems.

The contributions of this work are: (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 set, (iii) a framework that considers measurement and process noise, which improves the applicability to real systems with uncertainty, and (iv) numerical comparisons with (Hu et al., 2020) showing $10\times$ tighter accuracy in $\frac{1}{2}$ the computation time.

## 2 PRELIMINARIES

Consider a discrete-time linear time-varying system,

$$\mathbf{x}_{t+1} = A_t\mathbf{x}_t + B_t\mathbf{u}_t + \mathbf{c}_t + \boldsymbol{\omega}_t = f(\mathbf{x}_t;\pi); \quad \mathbf{y}_t = C_t^T\mathbf{x}_t + \boldsymbol{\nu}_t; \quad \mathbf{u}_t = \pi(\mathbf{y}_t) \tag{1}$$

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), and output-feedback controller $\pi$ is parameterized by an $m$-layer feed-forward NN (defined formally in Appendix B.1), optionally subject to control constraints, $\mathbf{u}_t \in \mathcal{U}_t$.

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

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 (1), 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. Furthermore, tight outer-approximations of the reachable sets enable verification of tight goal and avoid set specifications about the exact system, and they reduce the chances of verification being unsuccessful even if the exact system meets the specifications.

### 2.1 NEURAL NETWORK ROBUSTNESS VERIFICATION

A key step in computing reachable sets of the closed-loop system (1) with a NN control policy in a reasonable amount of time is to relax nonlinear constraints induced by the NN's nonlinear activation functions. This work leverages the following theorem to compute over-approximations of the reachable sets by extending the convex relaxation-based techniques in computer vision and machine learning literature which indicates within some known range of the input of a particular neuron, the nonlinear activation can be linearly bounded between upper/lower bounds.

**Theorem 2.1** (From (Zhang et al., 2018), 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*

$$\pi_j^U(\mathbf{y}) = \boldsymbol{\Lambda}_{j,:}^{(0)}\mathbf{y} + \sum_{k=1}^{m} \boldsymbol{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \boldsymbol{\Delta}_{:,j}^{(k)}); \quad \pi_j^L(\mathbf{y}) = \boldsymbol{\Omega}_{j,:}^{(0)}\mathbf{y} + \sum_{k=1}^{m} \boldsymbol{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \boldsymbol{\Theta}_{:,j}^{(k)}), \tag{2}$$

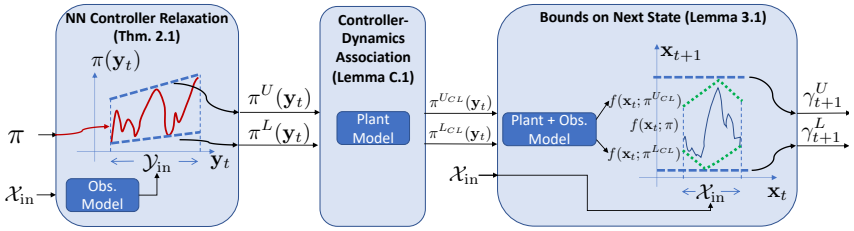



Figure 2. Approach Overview for simple 1D system. Theorem 2.1 relaxes the NN to compute the control boundaries $\pi^U, \pi^L$, given the observation $y_t$. Lemma C.1 uses the system dynamics to associate $\pi^U, \pi^L$ with the next state set. Lemma 3.1 compute bounds on the next state, $\gamma^U_{t+1}, \gamma^L_{t+1}$.

*where $\mathbf{\Lambda}, \mathbf{\Omega}, \mathbf{\Delta}, \mathbf{\Theta}$ are defined using NN weights and for general activations in (Zhang et al., 2018).*

In a closed-loop system, Theorem 2.1 provides a bound on 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 2.1 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 bounds on the NN output, using Corollary 3.3 of (Zhang et al., 2018).

We could leverage (Zhang et al., 2018) 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 (Xiang et al., 2020; Yang et al., 2019), 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 3.

## 3 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 relaxing the NN controller using Theorem 2.1, we associate linearized control bounds with extreme next states in Appendix C. Using the linearized extreme controller, we optimize over all states in the input set to find extreme next states in Section 3.2. We describe how to convert the solutions of the optimization problems into reachable set descriptions in Appendix E.1 then extend the formulation to handle control limits in Appendix E.3.

### 3.1 Assumptions

This work assumes that $\mathcal{X}_{\text{in}}$ is described by either (i) 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 (ii) 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 (a) an $\ell_\infty$ ball with radius $\epsilon$, s.t. $\mathcal{X}_{\text{out}} = \mathcal{B}_\infty(\mathbf{x}_0, \epsilon)$; or (b) 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 (cases (a) and (c)), or that $\mathbf{A}^{\text{out}} = \mathbf{I}_{n_x}$ (case (b)). Note that we use j, j, and $j$ to index the state vectors, polytope facets, and the control vectors respectively. Section 3.2 and Appendix C assume that $\mathcal{U}_t = \mathbb{R}^{n_u}$ (no control constraints); this assumption is relaxed in Appendix E.3.

### 3.2 Bounds on $\mathbf{x}_{t+1}$ from any $\mathbf{x}_t \in \mathcal{X}_{\text{IN}}$

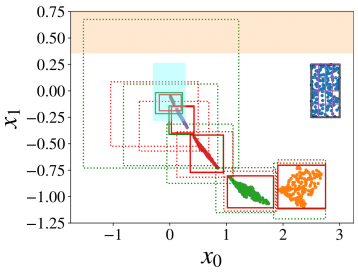
To compute "$t > 1$"-step reachable sets recursively, we form bounds on the next state polytope facet given a *set* of possible current states.

**Lemma 3.1.** *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 (1), and specification matrix $\mathbf{A}^{out} \in \mathbb{R}^{n_{out} \times n_x}$, for each $\mathrm{j} \in [n_{out}]$, there exist two fixed values $\gamma^U_{t+1,\mathrm{j}}$ and $\gamma^L_{t+1,\mathrm{j}}$ such that $\forall \mathbf{x}_t \in \mathcal{X}_{in}$, $\gamma^L_{t+1,\mathrm{j}} \leq \mathbf{A}^{out}_{\mathrm{j},:} f(\mathbf{x}_t; \pi) \leq \gamma^U_{t+1,\mathrm{j}}$ holds, where*

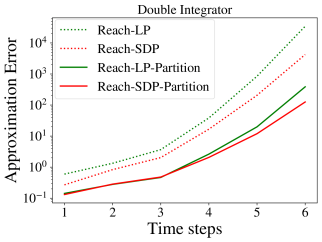
$$\gamma^U_{t+1,\mathrm{j}} = \max_{\mathbf{x}_t \in \mathcal{X}_{in}} \mathbf{M}^U_{\mathrm{j},:} \mathbf{x}_t + \mathbf{n}^U_{\mathrm{j}}; \quad \gamma^L_{t+1,\mathrm{j}} = \min_{\mathbf{x}_t \in \mathcal{X}_{in}} \mathbf{M}^L_{\mathrm{j},:} \mathbf{x}_t + \mathbf{n}^L_{\mathrm{j}}, \tag{3}$$

| Algorithm | Runtime [s] | Error |
|---|---|---|
| Reach-SDP (Hu et al., 2020) | 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

Figure 3. Reachable Sets for Double Integrator. In (a), Reach-LP is 30× faster to compute but 4× looser than Reach-SDP (Hu et al., 2020). 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 (Hu et al., 2020). 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).

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

$$\mathbf{M}^U_{\mathrm{j},:} = \left(\mathbf{A}^{out}_{\mathrm{j},:}\left(A_t + B_t \boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:} C_t^T\right)\right); \quad \mathbf{M}^L_{\mathrm{j},:} = \left(\mathbf{A}^{out}_{\mathrm{j},:}\left(A_t + B_t \boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:} C_t^T\right)\right) \tag{4}$$

$$\mathbf{n}^U_{\mathrm{j}} = \mathbf{A}^{out}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\left(\bar{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\nu}}_t + \underline{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \bar{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\omega}}_t + \underline{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\omega}}_t\right) \tag{5}$$

$$\mathbf{n}^L_{\mathrm{j}} = \mathbf{A}^{out}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:}\left(\underline{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\nu}}_t + \bar{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \underline{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\omega}}_t + \bar{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\omega}}_t\right), \tag{6}$$

*and where* $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$, $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$ *are defined as in Eqs.* (16) *and* (17)*, but using* $\mathbf{A}^{out}_{\mathrm{j},:} B_{t,:,\mathrm{j}} \boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:} C_t^T$ *and* $\mathbf{A}^{out}_{\mathrm{j},:} B_{t,:,\mathrm{j}} \boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:} C_t^T$*, respectively, with* $\boldsymbol{\Upsilon}, \boldsymbol{\Psi}, \mathbf{z}^U, \mathbf{z}^L, \bar{\mathbf{J}}, \underline{\mathbf{J}}$ *computed from Lemma C.1 (see Appendix).*

The optimization problems in (3) are linear programs that we solved with `cvxpy` (Diamond & Boyd, 2016), as opposed to the semi-definite programs in (Hu et al., 2020). Converting the state constraints from (3) into reachable sets is described in Appendix E.1, computing tighter reachable sets via parititioning the input set is in Appendix E.2, and the handling of control limits is in Appendix E.3.

## 4 Numerical Experiments

Consider a LTI double integrator system from (Hu et al., 2020) with a NN controller with [5,5] neurons and ReLU activations (see Appendix F.1 for more detail). Four algorithms are compared including Reach-SDP (Hu et al., 2020), Reach-LP, and those two algorithms after partitioning the input set into 16 cells. As shown in Fig. 3a, Reach-LP-Partition provides a 10× improvement in reachable set tightness over the prior state-of-the-art, Reach-SDP (Hu et al., 2020), while requiring $\frac{1}{2}$ of the computation time. Fig. 3b shows sampled trajectories, where each colored cluster of points represents sampled reachable states at a particular timestep (blue→orange→green, etc.). This figure also visualizes reachable set bounds for various algorithms. Recall that sampling trajectories could miss possible reachable states, whereas these algorithms are guaranteed to over-approximate the reachable sets, however they provide various degrees of tightness to the sampled points.

We quantitatively compare the rectangle sizes in Fig. 3b by defining the tightness as the ratio of areas between the smallest axis-aligned bounding box on the sampled points and the provided reachable set (minus 1). Fig. 3c shows as the system progresses forward in time, all algorithms' tightness get worse, but Reach-LP-Partition and Reach-SDP-Partition perform the best and similarly. Additional results in Appendix F include ablation, runtime analysis, and a 6D quadrotor example.

## 5 Conclusion

This paper proposed a convex relaxation-based algorithm for computing forward reachable sets of closed-loop systems with NN controllers. The prior work is limited to shallow NNs and is computationally intensive, which limits applicability to real systems. This work advances the state-of-the-art in guaranteeing properties of systems that employ NNs in the feedback loop. [1]

[1]This work was supported by Ford Motor Company.

## REFERENCES


Matthias Althoff. An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Erling D Andersen and Knud 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.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire 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.

Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pp. 258–263. Springer, 2013.

François Chollet et al. Keras. `https://keras.io`, 2015.

Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew 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.

Souradeep Dutta, Xin Chen, and Sriram 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.

Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *ATVA*, 2017.

Michael Everett, Golnaz Habibi, and Jonathan P How. Robustness analysis of neural networks via efficient partitioning with applications in control systems. *IEEE Control Systems Letters*, 2020.

Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. Automatic reachability analysis for nonlinear hybrid models with c2e2. In *International Conference on Computer Aided Verification*, pp. 531–538. Springer, 2016.

Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li, and Qi 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.

Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *arXiv preprint arXiv:1903.01287*, 2019.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pp. 379–395. Springer, 2011.

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.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George 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.

Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kunčak (eds.), *Computer Aided Verification*, pp. 3–29, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9.

Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup 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.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017a.

Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel 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, 2017b.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *CoRR*, abs/1706.07351, 2017. URL `http://arxiv.org/abs/1706.07351`.

Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, and Taylor T Johnson. Verification of closed-loop systems with neural network controllers. *EPiC Series in Computing*, 61:201–210, 2019.

Antonis Papachristodoulou and Stephen 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.*, volume 3, pp. 3482–3487. IEEE, 2002.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pp. 10802–10813, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019.

Claire J Tomlin, John Lygeros, and S Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S Boning, and Inderjit S Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)*, 2018.

Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5283–5292, 2018.

Weiming Xiang, Hoang-Dung Tran, Joel A Rosenfeld, and Taylor 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.

Weiming Xiang, Hoang-Dung Tran, Xiaodong Yang, and Taylor T Johnson. Reachable set estimation for neural network control systems: A simulation-guided approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Guoqing Yang, Guangyi Qian, Pan Lv, and Hong 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.

Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters*, 5(2):1127–1134, 2020.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pp. 4939–4948, 2018.

# SUPPLEMENTARY MATERIALS

## A 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 properties of NNs, often motivated by defending against adversarial examples (Szegedy et al., 2014). These methods broadly range from exact (Katz et al., 2017a) to tight (Fazlyab et al., 2019) to efficient (Zhang et al., 2018) to fast (Gowal et al., 2018). Although these tools are not designed for closed-loop systems, the NN relaxations from (Zhang et al., 2018) provide a key foundation for this work's analysis.

For closed-loop systems, reachability analysis is a standard component of safety verification. Modern methods include Hamilton-Jacobi Reachability methods (Tomlin et al., 2000; Bansal et al., 2017), SpaceEx (Frehse et al., 2011), Flow* (Chen et al., 2013), CORA (Althoff, 2015), and C2E2 (Duggirala et al., 2015; Fan et al., 2016), 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 (Papachristodoulou & Prajna, 2002) and control barrier functions (CBFs) (Ames et al., 2016).

Recent reachability analysis approaches that do account for NN control policies face a tradeoff between computation time and conservatism. (Dutta et al., 2019; Huang et al., 2019; Fan et al., 2020) use polynomial approximations of NNs to make the analysis tractable. Most works consider NNs with ReLU approximations, whereas (Ivanov et al., 2019) considers sigmoidal activations. (Xiang et al., 2020; Yang et al., 2019) introduce conservatism by assuming the NN controller could output its extreme values at every state. Most recently, (Hu et al., 2020) formulated the problem as a SDP, called Reach-SDP. This work builds on (Hu et al., 2020) and makes it more scalable by re-formulating the SDP as a linear program, introduces sources of uncertainty in the closed-loop dynamics, and shows that performance improvements can be achieved by partitioning the input set.

## B PRELIMINARIES

### B.1 CONTROL POLICY NEURAL NETWORK STRUCTURE

Using notation from (Zhang et al., 2018), 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 $\{1, 2, \ldots, i\}$. 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 (i.e., may need projection to $\mathcal{U}_t$),

$$\mathbf{u}_t = \pi(\mathbf{y}_t) = \Phi_m(\mathbf{y}_t) = \mathbf{W}^{(m)}\Phi_{m-1}(\mathbf{y}_t) + \mathbf{b}^{(m)}. \tag{7}$$

## C BOUNDS ON $\mathbf{x}_{t+1}$ FROM A PARTICULAR $\mathbf{x}_t$

**Lemma C.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 (1), and specification matrix $\mathbf{A}^{out} \in \mathbb{R}^{n_{out} \times n_x}$, for each $\mathrm{j} \in [n_{out}]$, there exist two explicit functions $\pi^{L_{CL}}_{:,\mathrm{j}} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ and $\pi^{U_{CL}}_{:,\mathrm{j}} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ 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 $\mathbf{A}^{out}_{\mathrm{j},:} f(\mathbf{x}_t, \pi^{L_{CL}}_{:,\mathrm{j}}) \leq \mathbf{A}^{out}_{\mathrm{j},:} f(\mathbf{x}_t, \pi) \leq \mathbf{A}^{out}_{\mathrm{j},:} f(\mathbf{x}_t, \pi^{U_{CL}}_{:,\mathrm{j}})$ holds true, where*

$$\pi^{U_{CL}}_{:,\mathrm{j}}(\mathbf{y}_t) = \boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\mathbf{y}_t + \mathbf{z}^U \tag{8}$$

$$\pi^{L_{CL}}_{:,\mathrm{j}}(\mathbf{y}_t) = \boldsymbol{\Xi}^{(0)}_{\mathrm{j},:,:}\mathbf{y}_t + \mathbf{z}^L, \tag{9}$$

*letting*

$$\mathbf{z}^U = \sum_{k=1}^{m} \left[ \mathbf{\Upsilon}^{(k)}_{\mathrm{j},:,:} \mathbf{b}^{(k)} + \mathbb{1}_{n_u} \left( \left( \mathbf{\Upsilon}^{(k)}_{\mathrm{j},:,:} \right)^T \odot \mathbf{\Psi}^{(k)} \right) \right] \tag{10}$$

$$\mathbf{z}^L = \sum_{k=1}^{m} \left[ \mathbf{\Xi}^{(k)}_{\mathrm{j},:,:} \mathbf{b}^{(k)} + \mathbb{1}_{n_u} \left( \left( \mathbf{\Xi}^{(k)}_{\mathrm{j},:,:} \right)^T \odot \mathbf{\Gamma}^{(k)} \right) \right] \tag{11}$$

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

$$\mathbf{\Upsilon}^{(k)}_{\mathrm{j},:,:} = \bar{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} \mathbf{\Lambda}^{(k)} + \underline{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} \mathbf{\Omega}^{(k)} \tag{12}$$

$$\mathbf{\Psi}^{(k)}_{\mathrm{j},:,:} = \mathbf{\Delta}^{(k)} \bar{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} + \mathbf{\Theta}^{(k)} \underline{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} \tag{13}$$

$$\mathbf{\Xi}^{(k)}_{\mathrm{j},:,:} = \bar{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} \mathbf{\Omega}^{(k)} + \underline{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} \mathbf{\Lambda}^{(k)} \tag{14}$$

$$\mathbf{\Gamma}^{(k)}_{\mathrm{j},:,:} = \mathbf{\Theta}^{(k)} \bar{\mathbf{J}}^{(k)}_{\mathrm{j},:,:} + \mathbf{\Delta}^{(k)} \underline{\mathbf{J}}^{(k)}_{\mathrm{j},:,:}. \tag{15}$$

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

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

$$\underline{\mathbf{J}}^{(k)}_{\mathrm{j},j,:} = \begin{cases} \mathbf{0}^T, & \textit{if } \mathbf{A}^{out}_{\mathrm{j},:} B_{t,:,\mathrm{j}} \geq 0 \\ \mathbf{e}_j^T, & \textit{otherwise} \end{cases}, \tag{17}$$

*and* $\mathbf{\Lambda}, \mathbf{\Omega}, \mathbf{\Delta}, \mathbf{\Theta}$ *are computed from Theorem 2.1 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 2.1, 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 $\pi^{U_{CL}}_{:,\mathrm{j}} \in \Pi(\mathbf{y}_t)$ as one that induces the least upper bound on the j-th facet of the next state polytope,

$$\begin{aligned} &\mathbf{A}^{\text{out}}_{\mathrm{j},:} f(\mathbf{x}_t; \pi^{U_{CL}}_{:,\mathrm{j}}) = \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}^{\text{out}}_{\mathrm{j},:} f(\mathbf{x}_t; \pi) \\ &= \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}^{\text{out}}_{\mathrm{j},:} \left[ A_t \mathbf{x}_t + B_t \pi(\mathbf{y}_t) + \mathbf{c}_t + \boldsymbol{\omega}_t \right] \\ &= \left[ \max_{\pi \in \Pi(\mathbf{y}_t)} \mathbf{A}^{\text{out}}_{\mathrm{j},:} B_t \pi(\mathbf{y}_t) \right] + \mathbf{A}^{\text{out}}_{\mathrm{j},:} \left[ A_t \mathbf{x}_t + \mathbf{c}_t + \boldsymbol{\omega}_t \right], \end{aligned} \tag{18}$$

Thus for $\mathbf{y}_t$,

$$\pi^{U_{CL}}_{:,\mathrm{j}} = \underset{\pi \in \Pi(\mathbf{y}_t)}{\operatorname{argmax}} \mathbf{A}^{\text{out}}_{\mathrm{j},:} B_t \pi(\mathbf{y}_t). \tag{19}$$

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

$$\pi^{U_{CL}}_{j,\mathrm{j}}(\mathbf{y}_t) = \begin{cases} \pi_j^U(\mathbf{y}_t), & \text{if } \mathbf{A}^{\text{out}}_{\mathrm{j},:} B_{t,:,j} \geq 0 \\ \pi_j^L(\mathbf{y}_t), & \text{otherwise} \end{cases}. \tag{20}$$

Writing (20) in matrix form results in (8). The proof of the lower bound follows similarly. □

## D PROOF OF LEMMA 3.1

**Lemma 3.1.** *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 (1), and specification matrix* $\mathbf{A}^{out} \in \mathbb{R}^{n_{out} \times n_x}$, *for each* $\mathrm{j} \in [n_{out}]$, *there exist two fixed values* $\gamma^U_{t+1,\mathrm{j}}$ *and* $\gamma^L_{t+1,\mathrm{j}}$ *such that* $\forall \mathbf{x}_t \in \mathcal{X}_{in}$, $\gamma^L_{t+1,\mathrm{j}} \leq \mathbf{A}^{out}_{\mathrm{j},:} f(\mathbf{x}_t; \pi) \leq \gamma^U_{t+1,\mathrm{j}}$ *holds, where*

$$\gamma^U_{t+1,\mathrm{j}} = \max_{\mathbf{x}_t \in \mathcal{X}_{in}} \mathbf{M}^U_{\mathrm{j},:} \mathbf{x}_t + \mathbf{n}^U_{\mathrm{j}}; \quad \gamma^L_{t+1,\mathrm{j}} = \min_{\mathbf{x}_t \in \mathcal{X}_{in}} \mathbf{M}^L_{\mathrm{j},:} \mathbf{x}_t + \mathbf{n}^L_{\mathrm{j}}, \tag{3}$$

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

$$\mathbf{M}^{U}_{\mathrm{j},:} = \left(\mathbf{A}^{out}_{\mathrm{j},:}\left(A_t + B_t \boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:} C_t^T\right)\right); \quad \mathbf{M}^{L}_{\mathrm{j},:} = \left(\mathbf{A}^{out}_{\mathrm{j},:}\left(A_t + B_t \boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:} C_t^T\right)\right) \tag{4}$$

$$\mathbf{n}^{U}_{\mathrm{j}} = \mathbf{A}^{out}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\left(\bar{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\nu}}_t + \underline{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \bar{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\omega}}_t + \underline{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\omega}}_t\right) \tag{5}$$

$$\mathbf{n}^{L}_{\mathrm{j}} = \mathbf{A}^{out}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:}\left(\underline{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\nu}}_t + \bar{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \mathbf{c}_t + \underline{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\omega}}_t + \bar{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\omega}}_t\right), \tag{6}$$

*and where* $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$, $\{\bar{\bar{\mathbf{J}}}, \underline{\bar{\mathbf{J}}}\}$ *are defined as in Eqs.* (16) *and* (17)*, but using* $\mathbf{A}^{out}_{\mathrm{j},:} B_{t,:,\mathrm{j}} \boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:} C_t^T$ *and* $\mathbf{A}^{out}_{\mathrm{j},:} B_{t,:,\mathrm{j}} \boldsymbol{\Psi}^{(0)}_{\mathrm{j},:,:} C_t^T$*, respectively, with* $\boldsymbol{\Upsilon}, \boldsymbol{\Psi}, \mathbf{z}^U, \mathbf{z}^L, \bar{\mathbf{J}}, \underline{\mathbf{J}}$ *computed from Lemma C.1 (see Appendix).*

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

$$\begin{aligned}
\mathbf{A}^{\text{out}}_{\mathrm{j},:}\mathbf{x}_{t+1} &= \mathbf{A}^{\text{out}}_{\mathrm{j},:} f(\mathbf{x}_t; \pi) && (21)\\
&\leq \mathbf{A}^{\text{out}}_{\mathrm{j},:} f(\mathbf{x}_t; \pi^{U_{CL}}_{:,\mathrm{j}}) && (22)\\
&\leq \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\mathrm{j},:} f(\mathbf{x}_t; \pi^{U_{CL}}_{:,\mathrm{j}}) := \gamma^U_{t+1,\mathrm{j}} && (23)\\
&= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\mathrm{j},:}\left[A_t\mathbf{x}_t + B_t\pi^{U_{CL}}_{:,\mathrm{j}}(\mathbf{y}_t) + c_t + \boldsymbol{\omega}_t\right] && (24)\\
&= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\mathrm{j},:}\left[A_t\mathbf{x}_t + B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\mathbf{y}_t + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right] && (25)\\
&= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \mathbf{A}^{\text{out}}_{\mathrm{j},:}\left[A_t\mathbf{x}_t + B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\left(C_t^T\mathbf{x}_t + \boldsymbol{\nu}_t\right) + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right] && (26)\\
&= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \left(\mathbf{A}^{\text{out}}_{\mathrm{j},:}\left(A_t + B_t\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}C_t^T\right)\right)\mathbf{x}_t + \\
&\qquad \mathbf{A}^{\text{out}}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\boldsymbol{\nu}_t + \mathbf{z}^U\right) + c_t + \boldsymbol{\omega}_t\right) && (27)\\
&= \max_{\mathbf{x}_t \in \mathcal{X}_{\text{in}}} \left(\mathbf{A}^{\text{out}}_{\mathrm{j},:}\left(A_t + B_t\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}C_t^T\right)\right)\mathbf{x}_t + \\
&\qquad \mathbf{A}^{\text{out}}_{\mathrm{j},:}\left(B_t\left(\boldsymbol{\Upsilon}^{(0)}_{\mathrm{j},:,:}\left(\bar{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\nu}}_t + \underline{\bar{\mathbf{J}}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\nu}}_t\right) + \mathbf{z}^U\right) + \right.\\
&\qquad \left. c_t + \bar{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\bar{\boldsymbol{\omega}}_t + \underline{\mathbf{J}}^{(0)}_{\mathrm{j},:,:}\underline{\boldsymbol{\omega}}_t\right), && (28)
\end{aligned}$$

where (25) substitutes the definition of $\pi^{U_{CL}}_{:,\mathrm{j}}$ from Lemma C.1, (26) substitutes the observation from (1), (27) separates terms that depend on $\mathbf{x}_t$, and (28) introduces the worst-case realizations of process and measurement noise. Substituting $\mathbf{M}^U_{\mathrm{j},:}, \mathbf{n}^U_{\mathrm{j}}$ results in (3). The proof of the lower bound follows similarly. □

# E Additional Components of Approach

## E.1 Converting State Constraints into Reachable Sets

### E.1.1 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

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \mathcal{B}_\infty\left(\frac{\gamma^U_{t+1,:} + \gamma^L_{t+1,:}}{2}, \frac{\gamma^U_{t+1,:} - \gamma^L_{t+1,:}}{2}\right). \tag{29}$$

### E.1.2 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 $(\gamma^L_{1,\text{j}}, \gamma^U_{1,\text{j}})$ for each index of output polytope facets $\text{j} \in [m_{out}]$, giving

$$\bar{\mathcal{R}}_{t+1}(\mathcal{X}_0) = \{\mathbf{x}_t | \begin{bmatrix} \mathbf{A}^{\text{out}} \\ -\mathbf{A}^{\text{out}} \end{bmatrix} \mathbf{x}_t \leq \begin{bmatrix} \gamma^U_{t+1,:} \\ -\gamma^L_{t+1,:} \end{bmatrix}\}. \tag{30}$$

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 (1).

### E.2 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 computing the reachable sets separately for subsets of $\mathcal{X}_0$, then taking the union of all reachable sets from each subset. This idea falls into the general framework from (Everett et al., 2020) of choosing a propagator (e.g., Reach-LP, Reach-SDP) and a partitioner (e.g., Uniform Gridding) for the analysis.

### E.3 Accounting for Control Limits, $\mathcal{U}_t$

The key terms in Lemma C.1 can be modified to account for control input constraints, as

$$\pi^{U_{CL}}_{:,\text{j}}(\mathbf{y}_t) = \text{Proj}_{\mathcal{U}_t}\left(\mathbf{\Upsilon}^{(0)}_{\text{j},:,:}\mathbf{y}_t + \mathbf{z}^U\right) \tag{31}$$

$$\pi^{L_{CL}}_{:,\text{j}}(\mathbf{y}_t) = \text{Proj}_{\mathcal{U}_t}\left(\mathbf{\Xi}^{(0)}_{\text{j},:,:}\mathbf{y}_t + \mathbf{z}^L\right), \tag{32}$$

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

$$\pi^{U_{CL}}_{j,\text{j}}(\mathbf{y}_t) = \begin{cases} \text{clip}(\pi^U_j(\mathbf{y}_t), \underline{\mathbf{u}}_j, \bar{\mathbf{u}}_j), & \text{if } \mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,j} \geq 0 \\ \text{clip}(\pi^L_j(\mathbf{y}_t), \underline{\mathbf{u}}_j, \bar{\mathbf{u}}_j), & \text{otherwise} \end{cases}, \tag{33}$$

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` (Diamond & Boyd, 2016) regardless). In this work, we only apply part of the control input constraint,

$$\pi^{U_{CL}}_{j,\text{j}}(\mathbf{y}_t) = \begin{cases} \min(\pi^U_j(\mathbf{y}_t), \bar{\mathbf{u}}_j), & \text{if } \mathbf{A}^{\text{out}}_{\text{j},:} B_{t,:,j} \geq 0 \\ \max(\pi^L_j(\mathbf{y}_t), \underline{\mathbf{u}}_j), & \text{otherwise} \end{cases}, \tag{34}$$

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 (Yang et al., 2020) of the clip function.

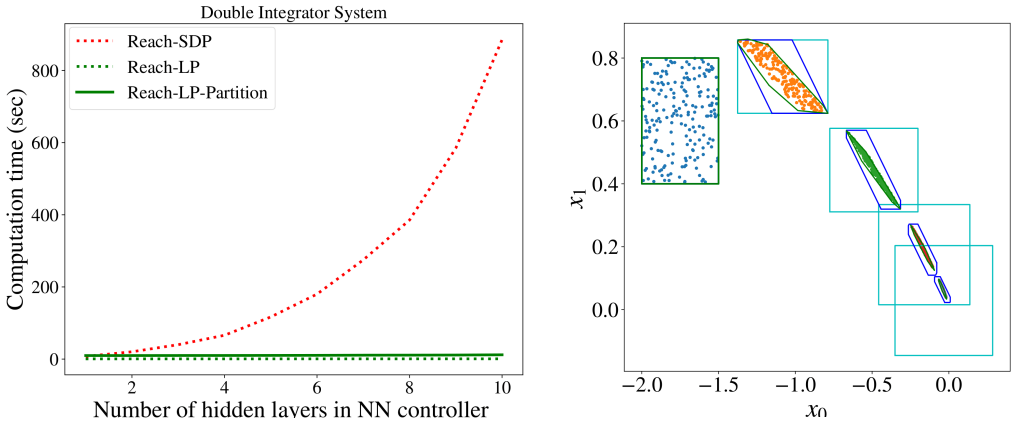
## F Additional Numerical Experiments

### F.1 Double Integrator

$$\mathbf{x}_{t+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A_t} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0.5 \\ 1 \end{bmatrix}}_{B_t} \mathbf{u}_t, \tag{35}$$

with $\mathbf{c}_t = 0$, $C_t = I_2$ and no noise, discretized with sampling time $t_s = 1s$. As in (Hu et al., 2020), 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 (Chollet et al., 2015) for 20 epochs with batch size 32.

We implemented Reach-SDP in Python with `cvxpy` and `MOSEK` (Andersen & Andersen, 2000). All computation times are reported from an i7-6700HQ CPU with 16GB RAM.




(a) Runtime with NN depth

(b) Number of Polytope Facets

Figure 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.

### F.2 Verification

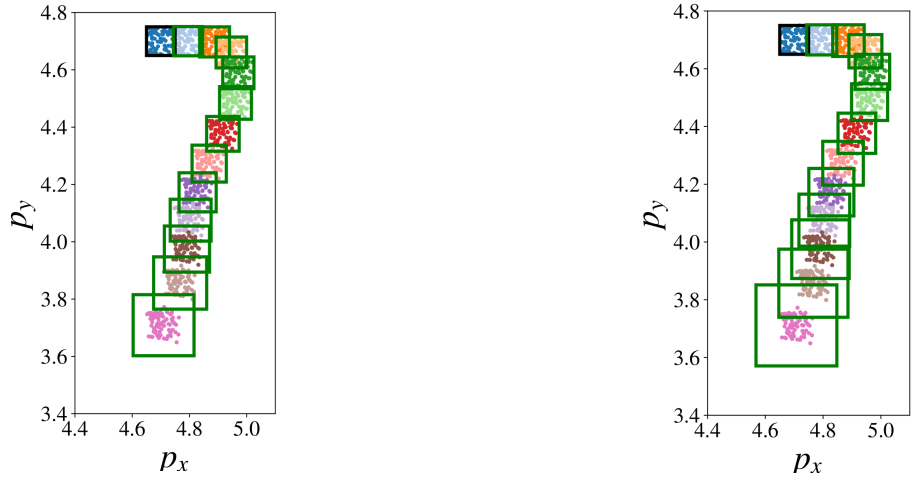
A primary application of reachable sets is to verify reach-avoid properties. In Fig. 3b, we consider a case with an avoid set $\mathcal{A} = \{\mathbf{x} | \mathbf{x}_1 \geq 0.35\}$ (orange) and a goal set $\mathcal{G} = [-0.25, 0.25] \times [-0.25, 0.25]$ (cyan). Each of the algorithms, except Reach-LP, are able to verify these properties for the 5-step scenario shown by computing the reachable sets and doing the set intersection operation outlined in Section 2. This highlights the importance of tight reachable sets.

### F.3 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.

### F.4 Ablation Study: $\ell_\infty$ vs. Polytopes

Recall that Appendix E.1 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 essentially a 4-polytope, and that $\mathcal{X}_0$ was chosen to show a different scenario than Fig. 3.




(a) No Noise

(b) Sensor & Process Noise

Figure 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), $\boldsymbol{\nu} \sim \texttt{Unif}(\pm 0.001 \cdot \mathbb{1}_6), \boldsymbol{\omega} \sim \texttt{Unif}(\pm 0.005 \cdot \mathbb{1}_6)$.

### F.5 6D Quadrotor with Noise

Consider the 6D nonlinear quadrotor from (Hu et al., 2020; Lopez et al., 2019),

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0_{3\times 3} & I_3 \\ 0_{3\times 3} & 0_{3\times 3} \end{bmatrix}}_{A_t} \mathbf{x}_t + \underbrace{\begin{bmatrix} & g & 0 & 0 \\ 0_{3\times 3} & 0 & -g & 0 \\ & 0 & 0 & 1 \end{bmatrix}^T}_{B_t} \underbrace{\begin{bmatrix} \tan(\theta) \\ \tan(\phi) \\ \tau \end{bmatrix}}_{\mathbf{u}_t} + \underbrace{\begin{bmatrix} 0_{5\times 1} \\ -g \end{bmatrix}}_{\mathbf{c}_t} + \boldsymbol{\omega}_t, \tag{36}$$

which differs from (Hu et al., 2020; Lopez et al., 2019) 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 (Hu et al., 2020), the state vector contains 3D positions and velocities, $[p_x, p_y, p_z, v_x, v_y, v_z]$, while nonlinearities from (Lopez et al., 2019) are absorbed into the control as functions of $\theta$ (pitch), $\phi$ (roll), and $\tau$ (thrust) (subject to the same actuator constraints as (Hu et al., 2020)). We implemented a similar nonlinear MPC as (Hu et al., 2020) 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 (36) 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.