# Stability analysis and controller synthesis using single-hidden-layer ReLU neural networks

Pouya Samanipour and Hasan A. Poonawala

***Abstract*—This paper presents algorithms to solve analysis and controller synthesis problems for dynamical systems modeled as a recurrent single-hidden-layer rectified linear unit neural network (ReLU NN), or equivalently, a piecewise affine dynamical system. Such models are interesting since they may arise through the use of modern machine learning methods for system identification, or as closed-loop solutions in certain classes of model predictive control (MPC) problems. A key idea in the proposed approach is to use piecewise affine Lyapunov functions parametrized as ReLU neural networks, and similarly parametrized controllers. This compatible representation between the Lyapunov function and the dynamics simplifies the automation of analysis of and controller synthesis for learned models. Our method of verifying a candidate Lyapunov function is faster than methods based on mixed integer programming. We 'learn' controllers and Lyapunov functions using both weight updates and network architecture search, without gradients. We demonstrate the proposed algorithm on examples involving learned models, explicit MPC controllers, and constrained controller synthesis.**

## I. Introduction

MODERN machine learning has made fitting complex functions to large amounts of data a realistic task to accomplish [1]. This ability is increasingly finding applications in control and dynamical systems. Two obvious applications are in system identification, where the right-hand side of the dynamics models are learned from data [2]–[7], and analysis, where Lyapunov functions are 'learned' [8]–[11].

If we are to increasingly model dynamical systems using over-parametrized models like neural networks, we require methods to analyze such models, and to possibly synthesize controllers from them. The potential complexity of such models suggests that the methods we develop must be amenable to automatic, and preferably efficient, computations.

Automatic methods for Lyapunov-based analysis and synthesis must deal with the fact that Lyapunov-based methods require solving a constraint satisfaction problem at infinite states [12]. One approach to doing so is to reformulate these infinite constraints involving states into finite constraints involving the parameters of the Lyapunov function and dynamical system. This method leads to (often convex) constrained optimization problems that yield valid Lyapunov functions [13]–[16] or controllers [15] when feasible. These methods are, however, parametrization-specific. Section II mentions related work in detail.

These optimization-based methods may be extended to piecewise versions [17] allowing analysis of complicated models. The work of automatically stitching together Lyapunov functions and stability conditions at the boundaries of these pieces has so far proved either algorithmically or computationally difficult [17], [18].

An alternative approach to dealing with infinite constraints is inspired by supervised machine learning [19], where a model that works on an infinite input set may be learned from a finite dataset. This idea leads to sampling-based methods for 'learning' Lyapunov functions [**?**], [8]–[11], [20]. Unlike standard machine learning, in Lyapunov analysis, we require that the stability conditions hold at all states, even when learned using finite states. This requirement forces the use of a *verification* step after training. Again, Section II reviews the literature in detail.

The advantage of the learn-then-verify approach using neural networks is that it can handle complex dynamics models and Lyapunov functions (like multi-layer neural networks) automatically. The disadvantage is that the verification step for most combinations of dynamics model and Lyapunov functions – if a step exists – is often expensive and returns only one out of possibly many counterexamples. Section II provides details.

In this paper, we focus on the stability analysis of continuous-time dynamical systems where the vector field is given by a single-hidden-layer rectified linear unit neural network (ReLU NN). Existing methods for computational Lyapunov analysis applicable to such systems focus on the binary nature of the ReLU activation. Formulating verification problems that explicitly feature these binary variables can lead to expensive algorithms. For example, the use of mixed integer linear programming is common, which is NP-hard.

The **main idea** in this paper is to exploit the continuous piecewise affine nature of ReLU NNs to design less expensive algorithms. We exploit the piecewise nature by preferring to enumerate the regions defined by the neurons, which will be fewer than the combinations one obtains when associating each neuron with a binary variable. We exploit the affine nature by using Farkas' lemma to eliminate quantifiers on the state without conservatism in the resulting conditions.

Pouya Samanipour and Hasan A. Poonawala are with the Department of Mechanical Engineering, University of Kentucky, Lexington, KY 40506, USA. samanipour.pouya,hasan.poonawala@uky.edu

This work was supported by the Department of Mechanical Engineering at the University of Kentucky.

### A. Contributions

1) We show how to formulate a single-hidden-layer rectified neural network (ReLU NN) as a piecewise affine function, and to check if it is a valid Lyapunov function using enumeration and linear programming. This procedure is fully automated. The main insight is to view the hidden layer neurons as a hyperplane arrangement.
2) We rigorously formulate the search for outer layer weights based on Lyapunov stability as a convex constrained optimization problem; the hidden layers are fixed. We use the inhomogenous form of Farkas' Lemma to eliminate quantifiers involving the state. This elimination is exact (lossless) and, therefore, less conservative compared to methods involving polynomial functions.
3) If stability conditions are violated in some states, we propose that one should add neurons to the Lyapunov function that partition the regions containing those states, and show how to choose them using new criteria. Thus, our method can consider increasingly expressive Lyapunov functions.
4) We extend the analysis method into a synthesis method for systems where the input enters linearly.

These innovations together lead to stability analysis and controller synthesis procedure that handles systems with 100s of regions in seconds. However, improvements are necessary to manage 1000s of regions, which easily occur in higher dimensions.

## II. Related Work

In this section, we review various approaches to automatically searching for Lyapunov functions, which can be grouped into the two categories below. We also briefly explain how our work relates to these approaches.

### A. Mathematical Optimization

Theorems of Alternatives [15], [21]–[23] are used to convert Lyapunov stability conditions involving quantifiers on the state into quantifier-free constraints. Which particular theorem is applied by a method depends on the parametrizations of the dynamical systems and Lyapunov function.

The sums-of-squares-based analysis is widely used for polynomial systems [24], [25]. A good amount of work focuses on homogeneous Lyapunov functions for homogeneous dynamical systems. [26]. Most remaining work has focused on stability analysis for switched and hybrid dynamical systems [27], [28]. These works include [16], [17], [29]–[31].

Common [27], multiple [32], or piecewise [29] quadratic Lyapunov functions are used to design switching rules [33]–[38] between affine or linear dynamics modes. Recent work in [18] reports that synthesis using PWQ Lyapunov functions may not be practical beyond a small number of cells.

Piecewise linear (PWL) Lyapunov functions were studied as an alternative to piecewise quadratic (PWQ) functions in [39] and [17]. A few methods attempt to find piecewise linear Lyapunov functions for piecewise affine systems [17], [40]–[42] and piecewise linear systems [43].

The ability to convert Lyapunov-based analysis of piecewise affine (PWA) dynamical systems into convex optimization problems has motivated attempts to automate the search for PWQ and PWL Lyapunov functions [17], [30], [31], [42], [44]. These works observe that an automated partitioning scheme may enable analysis of systems for which using the same partition as the dynamics for the PWQ or PWL Lyapunov function fails to produce a valid Lyapunov function. They use the optimal variables to identify which cells to refine and then divide these cells into two or more cells, typically along the longest edge. Despite this broad work in the analysis of PWA systems, only recently has their representation using ReLU neural networks been explored (see Section II-B).

### B. Learning From Finite Samples

Several methods are inspired by the intuitive idea that if we make a candidate Lyapunov function satisfy stability conditions at a sufficient number of finite state samples, the stability conditions may be satisfied at all points in a region of interpolation between these samples. To ensure that this 'generalization' happens, these approaches search for a counterexample, a state where the conditions fail, hoping that none will be found. The differences between various learning-from-samples methods lie in what tools are available to implement a search for a counterexample and how this counterexample is used.

The most common approach for finding a counterexample is to find the state at which the derivative of the Lyapunov function is maximum. If this derivative is greater than zero, then the state is a counterexample. For ReLU NNs and piecewise affine dynamical systems, this search is often implemented as a mixed integer linear program [43], [45]. Instead of manually derived optimization problems that drive the automatic search for a counterexample, some research directions use satisfiability modulo theory (SMT) solvers [46], [47]. These methods have been applied to dynamics that are piecewise constant [48], set-valued [49], nonlinear [9], [50], [51] or uncertain dynamics [5].

One approach for using the counterexample is to add it to the training samples [8]–[10], [45], [50] and repeat the learning process. A second approach uses the counterexample to prune the set of parameters in which a valid Lyapunov function may be found [11], [52], [53]. Either a Lyapunov function is found in the remaining set, or the set is pruned to emptiness, and no Lyapunov function exists for the system in the considered class.

## III. Problem Description And Paper Outline

Consider a controlled dynamical system of the form

$$\dot{x} = \mathrm{NN}_{\mathrm{ReLU}}(x, \theta'_x) + Bu, \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta'_x)$ denotes a rectified neural network with parameters $\theta'_x$, and $u \in \mathbb{R}^p$ is the control input. Such models may be learned from trajectory data or an existing dynamics model in a different parametrization. Section IV provides details on rectified neural networks.

A controller for (1) of the form $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_u)$ will yield a closed-loop of the form

$$\dot{x} = \mathrm{NN}_{\mathrm{ReLU}}(x, \theta_x). \tag{2}$$

We would like to choose parameters $\theta_u$ of the controller so that the origin of (2) is asymptotically stable, or simply analyze systems for which dynamics and controller parameters are given. Furthermore, we focus on Lyapunov functions that are also rectified neural networks, because they afford development of automated algorithms. These goals lead to three problems:

1) **Problem 1**: Verify whether a given candidate Lyapunov function of the form $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_V)$ is a valid function certifying asymptotic stability of the origin of (2).
2) **Problem 2**: Find a valid Lyapunov function of the form $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_V)$ certifying asymptotic stability of the origin of (2).
3) **Problem 3**: Find a state-feedback law of the form $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_u)$ that asymptotically stabilizes (1).

In section IV we highlight that rectified neural networks are continuous piecewise functions. Section V uses this fact to formulate a solution to Problem 1 (verification), based on linear programming. Section VI formulates the Lyapunov conditions into equivalent conditions that do not have quantifiers on the state. Section VII uses these conditions to propose an algorithm that solves Problem 2 (search), and Section VIII extends it further to propose a solution to Problem 3 (controller synthesis). Finally, we present examples in Section IX, discuss limitations in Section X, and conclude the paper.

## IV. Rectified Neural Networks As Continuous Piecewise Affine Functions

This section develops the notation we will use when formulating stability conditions. The main point is that rectified neural networks are continuous piecewise affine functions, and we will formulate stability conditions by converting the former to the latter. We refer to the standard representation of piecewise affine functions (see Section IV-B) as an **explicit parametrization**, while single-hidden-layer ReLU NNs (see Section IV-C) are an **implicit parametrization** of (continuous) piecewise affine functions.

**Notation:** The indices of elements of a set $S$ form the set $I(S)$. We use the bold symbol $\mathbf{x}_S$ to denote a set of variables $\{x_i\}_{i \in I(S)}$, and $\mathbf{x}_I$ to denote a set of variables $\{x_i\}_{i \in I}$.

We denote the convex hull of $S$ by $\mathrm{conv}\,(S)$, the interior of $S$ by $\mathrm{Int}\,(S)$, the boundary of $S$ by $\partial S$, and closure of $S$ by $\overline{S}$. The convex closure $\overline{\mathrm{conv}}\,(S)$ is simply $\overline{\mathrm{conv}\,(S)}$. Let $B_\epsilon(x) = \{y \in R^n : \|y - x\| < \epsilon$, an $\epsilon$-ball at $x$.

Let $I$ be a set of indices. Given a matrix $E$, $E^I$ denotes a matrix formed by stacking the $i^{\mathrm{th}}$ rows of $E$, for $i \in I$. The transpose of matrix $A$ is $A^T$.

For $v, u \in \mathbb{R}^n$, $v \succeq u \iff v_i \geq u_i$ for $1 \leq i \leq n$. The symbols $\preceq$, $\succ$, and $\prec$ imply the same element-wise rule corresponding to $\leq$, $>$, and $<$ respectively.

Let $\mathcal{B}^m \subset \mathbb{R}^m = \{-1, 1\}^m$, the set of $m$-dimensional vectors whose elements are $-1$ or 1. The vector $\mathbf{1}_m \in \mathcal{B}^m$ has all elements equal to unity. The vector $\mathbf{0}_m \in \mathbb{R}^m$ has all elements equal to zero.

### A. Partitions and Refinements

A partition $\mathcal{P}$ is a collection of subsets $\{X_i\}_{i \in I(\mathcal{P})}$, where $X_i \subseteq \mathbb{R}^n$, $n \in \mathbb{N}$, and $\overline{\mathrm{Int}\,(X_i)} = X_i$ for each $i \in I(\mathcal{P})$. Furthermore, $\mathrm{Int}\,(X_i) \cap \mathrm{Int}\,(X_j) = \emptyset$ for each pair $i, j \in I(\mathcal{P})$ such that $i \neq j$. We refer to $\cup_{i \in I(\mathcal{P})} X_i$ as the domain of $\mathcal{P}$, which we also denote by $\mathrm{Dom}(\mathcal{P})$. We also refer to the subsets $X_i$ in $\mathcal{P}$ as the cells of the partition. We assume that there exists a neighborhood of $x$ that intersects with only a finite number of cells in $\mathcal{P}$, for each $x \in \mathrm{Dom}(\mathcal{P})$.

Let $\mathcal{P} = \{Y_i\}_{i \in I}$ and $\mathcal{R} = \{Z_j\}_{j \in J}$ be two partitions of a set $S = \mathrm{Dom}\,(\mathcal{P}) = \mathrm{Dom}\,(\mathcal{R})$. A partition $\mathcal{R}$ is a *refinement* of $\mathcal{P}$ if $Z_j \cap Y_i \neq \emptyset$ implies that $Z_j \subseteq Y_i$. We denote the set of refinements of a partition $\mathcal{P}$ as $\mathrm{Ref}(\mathcal{P})$.

### B. Piecewise Affine Functions

We parameterize a piecewise affine function $\mathrm{PWA}(x)$ by a partition $\mathcal{P} = \{X_i\}_{i \in I(\mathcal{P})}$ and a collection of matrices $\mathbf{A}_{\mathcal{P}} = \{A_i\}_{i \in I(\mathcal{P})}$ and vectors $\mathbf{a}_{\mathcal{P}} = \{a_i\}_{i \in I(\mathcal{P})}$, where $X_i = \{x \in \mathbb{R}^n : E_i x + e_i \succeq 0\}$. We may then explicitly write $\mathrm{PWA}(x)$ as

$$\mathrm{PWA}(x) = A_i x + a_i, \text{ if } E_i x + e_i \succeq 0. \tag{3}$$

Note that a generic piecewise affine function may not be continuous unless we appropriately constrain the parameters $A_i$, $a_i$, $E_i$, and $e_i$ [17], [40].

### C. Single-Hidden-Layer ReLU NNs

A single-hidden-layer ReLU neural network with parameters $\theta = (W, H, b, c)$ defines a map from $x \in \mathbb{R}^n$ to output $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta) \in \mathbb{R}^o$:

$$\mathrm{NN}_{\mathrm{ReLU}}(x, \theta) = W\sigma_+.(Hx + b) + c, \tag{4}$$

where $W \in \mathbb{R}^{o \times m}$, $H \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^{o \times 1}$, and the function $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function. We choose the activation as the rectified linear activation $\sigma_+$ given by

$$\sigma_+(x) = \max(x, 0). \tag{5}$$

The period in the notation $\sigma_+.(v)$ in (4) implies that we apply this function to a vector $v \in \mathbb{R}^m$ in an element-wise manner to produce another vector in $\mathbb{R}^m$.

The pair $(H^i, b^i)$ – the $i^{\mathrm{th}}$ row of $H$ and the $i^{\mathrm{th}}$ element of $b$ – together represent a single neuron in the hidden layer of the ReLU neural network with parameter $\theta$ . The reflection of a neuron $(H^i, b^i)$ is the neuron $(-H^i, -b^i)$.

We refer to the pair $(H, b)$ as the (hidden) **neurons** of the network. The matrix $W$ and vector $c$ denote the **outer layer weights**, since they define the linear combination of the hidden layer outputs that produce the output of the network.

To distinguish between networks, we may add a subscript to the parameters of a network. For example, $\theta_x$ which corresponds to $(W_x, H_x, b_x, c_x)$.

Consider two networks with parameters $\theta_1$ and $\theta_2$ respectively that have the same input dimension but possibly different output dimensions, say $o_1$ and $o_2$. We define the weighted combination $\theta_3$ of the two networks as

$$\theta_3 = M_1\theta_1 \oplus M_2\theta_2 = (W_3, H_3, b_3, c_3), \tag{6}$$

where $M_1 \in \mathbb{R}^{o_3 \times o_1}$ and $M_2 \in \mathbb{R}^{o_3 \times o_2}$ for some $o_3 \in \mathbb{N}$, and

$$W_3 = \begin{bmatrix} M_1 W_1 & M_2 W_2 \end{bmatrix}, \; c_3 = M_1 c_1 + M_2 c_2, \tag{7}$$

$$H_3 = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \text{ and } b_3 = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{8}$$

We use this operation to define a neural network that is the sum of the output of two other networks or to insert the neurons of one network into another.

## D. The Partition Due To Neurons

The neurons $(H, b)$ of a single-hidden-layer rectified neural network $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta)$ define a hyperplane arrangement in $\mathbb{R}^n$, and equivalently a partition of $\mathbb{R}^n$ into polyhedra. We denote this partition as $\mathcal{P}_\theta$.

Given parameters $H \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, we define the function $\mathrm{sv}^\theta \colon \mathbb{R}^n \to \mathbb{R}^m$ as

$$\mathrm{sv}^\theta(x) = \mathrm{sign}.(Hx + b), \tag{9}$$

where sign is the usual function with range $\{1, 0, -1\}$, and the period again denotes element-wise application of a function to a vector. This function is similar to the sign vector defined in [54]. It is piecewise constant over $\mathbb{R}^n$.

Consider a relation $R$ on $X$ given by

$$x_1 R x_2 \implies \mathrm{sv}^\theta(x_1) = \mathrm{sv}^\theta(x_2)$$

This relation can be shown to be an equivalence relation. Let $\mathcal{G}_\theta = \{Y_k\}_{k \in I(\mathcal{G}_\theta)}$ be the set of equivalence classes of $X$ under $R$. This construction makes $\mathcal{G}_\theta$ a simplicial complex; it is not, however, a partition in our sense.

The partition $\mathcal{P}_\theta$ will be formed by the closure of the open cells of $\mathcal{G}_\theta$. We can collect these open cells by noting that every entry of $\mathrm{sv}^\theta(x)$ for $x$ in such a cell must be non-zero. In other words, the points in these cells do not lie in any of the hyperplanes defined by $(H, b)$. Consider the set $\kappa(\mathcal{G})$ of indices given by

$$\kappa(\mathcal{G}_\theta) = \{k \in I(\mathcal{G}_\theta) \colon x \in Y_k \implies (\mathrm{sv}^\theta(x))_i \neq 0 \forall\, 1 \leq i \leq m\}.$$

Then,

$$\mathcal{P}_\theta = \{\overline{Y_k}\}_{k \in \kappa(\mathcal{G}_\theta)}. \tag{10}$$

Henceforth, we will refer to the $i^{\text{th}}$ element of $\mathcal{P}_\theta$ as $X_i$. To each $X_i$ we can associate the vector $\mathrm{sv}_i^\theta \in \mathcal{B}^m$, which is the vector $\mathrm{sv}^\theta(x)$ for any $x \in \mathrm{Int}\,(X_i)$.

Note that constructing elements of partition $\mathcal{P}_\theta$ through the closure of open cells of $\mathcal{G}_\theta$ ensures that even points in the closed cells of $\mathcal{G}_\theta$ belong to $\mathrm{Dom}\,(\mathcal{P}_\theta)$.

## E. Converting $\mathrm{NN}_{\mathrm{ReLU}}(\mathrm{x})$ to $\mathrm{PWA}(\mathrm{x})$

We now represent $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta)$ as a piecewise affine function using the partition $\mathcal{P}_\theta$ and the vector $\mathrm{sv}_i^\theta$ associated with each cell $X_i \in \mathcal{P}_\theta$.

We convert each $\mathrm{sv}_i^\theta \in \mathcal{B}^m$ into two diagonal matrices $S_i^\theta, D_i^\theta \in \mathbb{R}^{m \times m}$, whose $j^{\text{th}}$ diagonal terms are given by

$$\left(S_i^\theta\right)_{jj} = \left(\mathrm{sv}_i^\theta\right)_j \text{ and}$$
$$\left(D_i^\theta\right)_{jj} = \sigma_+\left(\left(\mathrm{sv}_i^\theta\right)_j\right)$$

respectively, where $\sigma_+ = \max(x, 0)$.

We can now describe all points satisfying in the polyhedron $X_i \in \mathcal{P}_\theta$ as

$$X_i = \{x \in \mathbb{R}^n \colon S_i^\theta H\ x + S_i^\theta b \succeq 0\}, \tag{11}$$

and therefore the function $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta)$ as an explicit piecewise affine function over the partition $\mathcal{P}_\theta$:

$$\mathrm{NN}_{\mathrm{ReLU}}(x, \theta) = W D_i^\theta H\ x + W D_i^\theta b + c, \quad \text{if } S_i^\theta H\ x + S_i^\theta b \succeq 0. \tag{12}$$

The equation has the same form as (3).

## V. ReLU Neural Networks As Candidate Lyapunov Functions

We use candidate Lyapunov functions of the form

$$V(x) = \mathrm{NN}_{\mathrm{ReLU}}(x, \theta_V) = w_V \sigma_+ \left(H_V x + b_V\right) + c_V. \tag{13}$$

A candidate Lyapunov function must be positive definite [12], i.e., $V(0) = 0$ and $V(x) > 0$ when $x \neq 0$. To be a valid Lyapunov function certifying asymptotic stability of the origin of a system, this candidate function must strictly decrease along all solutions of the system (excluding the origin) [12].

Several methods check if these conditions hold at all states given a particular choice of parameters for the Lyapunov function. These methods suggest using mixed integer linear programming (MILP) or formal verification (SMT solvers). These methods are computationally expensive.

This section argues that working with the piecewise nature of the ReLU NN function is a better approach. Specifically, enumerating all pieces and using linear programming to verify the desired conditions can be effective.

## A. The Partition Associated With Stability Conditions

The dynamics function $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_x)$ and $\mathrm{NN}_{\mathrm{ReLU}}(x, \theta_V)$ define (possibly different) partitions $\mathcal{P}_{\theta_x}$ and $\mathcal{P}_{\theta_V}$. The Lyapunov functions and dynamics vector field may not be affine functions over the cells of either partition. However, they will be affine functions for cells in a partition that is a common refinement of both partitions.

We can redefine both functions to have a partition identical to this common refinement by combining their neurons, by redefining the parameters $\theta_x$ and $\theta_V$ to be

$$\theta_x \leftarrow \theta_x \oplus \mathbf{0}_n \theta_V, \tag{14}$$

$$\theta_V \leftarrow \mathbf{0}_n^T \theta_x \oplus \theta_V. \tag{15}$$

These redefined parameters now have the same neurons, but the redefined outer layer weights ensure that corresponding networks still produce the same output as before (see Section IV-C). Under this redefinition $\mathcal{P}_{\theta_x} = \mathcal{P}_{\theta_V}$.

For the rest of this paper, we assume that $\theta_x$ and $\theta_V$ have been redefined to have identical partitions. We will refer to this common partition as $\mathcal{P}_{\theta_V}$, which has an index set $I_{\theta_V}$.

## B. Stability Conditions For A Single Cell

When $V(x)$ is differentiable at $x$, and the dynamics $\dot{x} = f(x)$ are continuous, stability may be assessed through the condition that the Lie derivative $\mathcal{L}_f V$ of $V$ along $f(x)$ be negative definite or negative semi-definite for all $x$ in some region $X \ni 0$:

$$\mathcal{L}_f V = \langle \nabla V, f(x) \rangle < 0, \tag{16}$$

where $\nabla V$ is the gradient of $V(x)$ and $\langle \cdot, \cdot \rangle$ is the usual inner product.

At a state $x$ where $V(x)$ is differentiable, let the affine Lyapunov function be $V(x) = p^T x + q$, and the dynamics be $\dot{x} = Ax + a$. The Lyapunov stability condition (16) at this state is

$$p^T(Ax + a) < 0. \tag{17}$$

Since the Lyapunov functions and dynamics we consider are piecewise affine, the parameters involved in condition (17) are identical for all states in connected cells.

These parameters, using (12), for each $i \in I_{\theta_V}$ are:

$$p_i^T = w_V D_i^{\theta_V} H_V, \tag{18}$$
$$q_i = w_V D_i^{\theta_V} b_V, \tag{19}$$
$$A_i = W_x D_i^{\theta_V} H_V, \text{ and} \tag{20}$$
$$a_i = W_x D_i^{\theta_V} b_V. \tag{21}$$

The dynamics and Lyapunov function for $x \in X_i$ are

$$V(x) = p_i^T x + q_i, \text{ and}$$
$$\dot{x} = A_i^T x + a_i.$$

If a cell $X_i$ is defined by the constraints $E_i x + e_i \geq 0$, then we can compactly represent the condition at all states in $X_i$ through the quantified condition

$$E_i x + e_i \succeq 0, x \neq 0 \implies p_i^T(A_i x + a_i) < 0. \tag{22}$$

Similarly, ensuring positivity of $V(x)$ on $X_i$ leads to condition

$$E_i x + e_i \succeq 0, x \neq 0 \implies p_i^T x + q_i > 0. \tag{23}$$

For each cell, we can check (22) and (23) using linear programming. Specifically, we would compute the maximum of $p_i^T(A_i x + a_i)$ and minimum of $p_i^T x + q_i$ over the set $\{x \in \mathbb{R}^n : E_i x + e_i \succeq 0\}$, and compare these optimum values to zero.

Note that (22) is not enough by itself for $x \in \partial X_i$, due to the non-differentiability of $V(x)$ there. Theorem 4 shows that all required conditions are eventually checked when checking (22) for all cells in $\mathcal{P}_{\theta_V}$.

## C. Enumerating All Cells In A Partition

We use the incremental enumeration algorithm from [55] to enumerate all the vectors $\mathrm{sv}_i^\theta$ created by $(H_V, b_V)$. This algorithm was shown to take less time than the well-known reverse search algorithm [56]. The algorithm in [55] potentially requires solving a large number of linear programs. A key step in ensuring low running times for this enumeration is to avoid allocating new memory for each instance of the linear program, and to instead reuse the memory allocated for the first linear program.

Let the complexity of solving a linear program in $d$ variables and $\mu$ constraints be $\mathrm{lp}(\mu, d)$. Consider $m$ neurons in $\mathbb{R}^n$. The potential number of cells in such an arrangement [55] is

$$\zeta(m, n) = \sum_{i=0}^{n} \binom{m}{i}. \tag{24}$$

Let $\mathcal{C}_m$ be the set of cells in any given arrangement. The complexity of enumerating these cells using the incremental enumeration algorithm in [55] is $\mathcal{O}(|\mathcal{C}_m| m \mathrm{lp}(m, n))$.

## D. Complexity of Verifying A Candidate Lyapunov Function

For each cell, $X_i \in \mathcal{P}_{\theta_V}$, the complexity of implementing each of the two linear programs in Section V-C is $\mathrm{lp}(m, n)$. Given that we will enumerate $\mathcal{C}_m$ cells, the complexity for verifying the Lyapunov candidacy and stability conditions is $\mathcal{O}(|\mathcal{C}_m| \mathrm{lp}(m, n))$.

## VI. Quantifier-Free Lyapunov Stability Conditions

In this section, we present sufficient conditions on the parameters $\theta_V = (w_V, H_V, b_V, c_V)$ that determine when a given function is a candidate Lyapunov function, and when a candidate Lyapunov function is a valid Lyapunov function certifying asymptotic stability of the origin. We assume that cells of $\theta_V$ is combined with $\theta_x$ using the procedure in Section V-A). Section VII-D describes modifications that verify other closed-loop properties.

A key idea is that we remove the quantifiers in (22) and (23). This step allows us to check conditions over multiple regions simultaneously, without resorting to mixed integer formulations. Our quantifier-free conditions are equivalent to the original stability conditions. By contrast, quantifier-free conditions derived from applying the $S$-Lemma [22] to piecewise quadratic Lyapunov functions are sufficient but not necessary [17]. Note that we do not claim that our conditions are necessary for verifying asymptotic stability.

## A. Positive Definiteness of V(x)

The condition $V(0) = 0$ is equivalent to

$$w_V \max .(b_V, 0) + c_V = 0 \tag{25}$$

To ensure that $V(x)$ is positive definite, we require that no cell contains the origin in its interior, otherwise the affine function for such a cell must be negative in a neighborhood of the origin. We can do so by ensuring that $H_V$, $b_V$ contains $n$ independent hyperplanes passing through the origin, and their reflections. We distinguish between cells that contain $x = 0$ on the boundary and those that do not, similar to the approach by Johansson in [17]. Let

$$I_{\theta_V}^{lin} = \{i \in I_{\theta_V} : 0 \in \partial X_i\}, \tag{26}$$
$$I_{\theta_V}^{aff} = \{i \in I_{\theta_V} : 0 \notin \partial X_i\}. \tag{27}$$

Depending on whether a cell contains the origin on the boundary or not, we use the following results to ensure the sign definiteness of such explicit piecewise affine functions.

**Lemma 1** (Lemma 4.7 [17]). *The following are equivalent*

1) $Ex \succeq 0, Ex \neq 0 \implies p^T x > 0$.
2) $\exists v \succ 0$ *such that* $E^T v = p$.

To convert the conditional statement $Gx + g \succeq 0 \implies p^T x + q \leq 0$ into a condition without quantifiers, we use the following result shown in [40]:

**Lemma 2** (Lemma 8 [40]). *Let the set* $\{x \in \mathbb{R}^n : Gx + g \succeq 0\}$ *be non-empty, where* $G \in \mathbb{R}^{l \times n}$, $g \in \mathbb{R}^l$ *for some* $l \in \mathbb{N}$. *Let* $p \in \mathbb{R}^n$ *and* $q \in \mathbb{R}$. *Then, the following are equivalent*

1) $Gx + g \succeq 0 \implies p^T x + q \leq 0$.
2) $\exists v \in \mathbb{R}^l$, $v \succeq 0$ *such that* $G^T v + p = 0$ *and* $g^T v + q \leq 0$.

When we apply these results to (13), we use its corresponding explicit representation as a piecewise affine function, similar to (3) or (12):

$$V = w_V D_i^{\theta_V} (H_V x + b_V) \text{ if } S_i^{\theta_V} H_V x + S_i^{\theta_V} b_V \succeq 0.$$

The resulting combined set of conditions for positivity of $V(x)$ when $x \neq 0$ leads to the following result.

**Lemma 3.** *Consider a function* $V(x)$ *as defined in* (13). *If there exist* $\mu_i \in \mathbb{R}^m$ *for each* $i \in I_{\theta_V}$ *such that*

$$\mu_i^T S_i^{\theta_V} H_V - w_V D_i^{\theta_V} H_V = 0, \quad \forall i \in I_{\theta_V}^{lin}, \tag{28}$$

$$\mu_i \succ 0, \quad \forall i \in I_{\theta_V}^{lin}, \tag{29}$$

$$\mu_i^T S_i^{\theta_V} H_V - w_V D_i^{\theta_V} H_V = 0, \quad \forall i \in I_{\theta_V}^{aff}, \tag{30}$$

$$\mu_i^T S_i^{\theta_V} b_V - w_V D_i^{\theta_V} b_V \leq 0, \quad \forall i \in I_{\theta_V}^{aff}, \tag{31}$$

$$\mu_i \succeq 0, \quad \forall i \in I_{\theta_V}^{aff}, \tag{32}$$

*then* $V(x)$ *is strictly positive for* $x \neq 0$.

*Proof.* This result is a straightforward application of Lemmas 1 and 2 to a strict positivity constraint applied piecewise for all the regions in partition $\mathcal{P}_{\theta_V}$ defined by the ReLU Lyapunov NN with parameters $\theta_V$. ■

## *B. Lyapunov Stability Conditions For Multiple Cells*

Given parameters $\theta_x$ and $\theta_V$, we may obtain a set of conditions that verify asymptotic stability of a dynamical system by combining conditions that determine the candidacy of the Lyapunov function with the conditions that determine the decrease of the Lyapunov function along solutions, for each cell. We assume the neurons of the dynamics and Lyapunov function have been combined as Section V-A describes.

Combining the quantifier-free version of condition (22) for all cells in $\mathcal{P}_{\theta_V}$, where the constants are given in (18)-(21) leads to our main result.

**Theorem 4.** *Consider a dynamical system of the form* (2), *and a candidate Lyapunov ReLU neural network function* $V(x)$ *of the form* (13). *We assume that the neurons of the associated networks have been combined as described in Section V-A, and use variables defined in* (18)-(21). *If the parameters* $\theta_V$ *of* $V(x)$ *satisfy conditions* (25), (28)-(32), *and the following conditions*

$$H_V^T S_i^{\theta_V} \nu_i + A_i^T p_i = 0, \quad \forall i \in I_{\theta_V}^{lin}, \tag{33}$$

$$\nu_i \succ 0, \quad \forall i \in I_{\theta_V}^{lin}, \tag{34}$$

$$H_V^T S_i^{\theta_V} \nu_i + A_i^T p_i = 0, \quad \forall i \in I_{\theta_V}^{aff}, \tag{35}$$

$$b_V^T S_i^{\theta_V} \nu_i + a_i^T p_i < 0, \quad \forall i \in I_{\theta_V}^{aff}, \text{ and} \tag{36}$$

$$\nu_i \succeq 0, \quad \forall i \in I_{\theta_V}^{aff}, \tag{37}$$

*then the origin* $x = 0$ *is asymptotically stable.*

*Proof.* Condition (25) ensures that $V(0) = 0$ and conditions (28)-(32) ensure that $V(x)$ is positive definite (see Lemma 3). Together, they ensure that $V(x)$ is a candidate Lyapunov function.

We must show that conditions (33)-(37) imply that the Lyapunov function decreases along solutions of the dynamical system (2).

For points in the interior of cells $X_i \in \mathcal{P}_{\theta_V}$, this decrease is given by the Lie derivative $L_f V$ which is equal to an affine function of the form on the left-hand side of (17). If $i \in I_{\theta_V}^{lin}$ and conditions (33) and (34) are satisfied, then condition (22) is true for cell $X_i$. Similarly if $i \in I_{\theta_V}^{aff}$ and conditions (35)-(37) are satisfied, then condition (22) is true for cell $X_i$. Condition (22) being true implies that the Lie derivative is negative at all points in $\operatorname{Int}(X_i)$.

For points on the boundary of cells, the gradient of $V(x)$ is not defined and so the Lie derivative is not defined. Instead, we must use the set-valued versions of these quantities which are the Clarke generalized gradient [57]–[59] and Clarke generalized derivative respectively [58], [60]. These objects exist for the case where the Lyapunov function and dynamics are parametrized as ReLU NNs, since they are Lipschitz continuous. As shown in Proposition 3.2 in [59], the rate of decrease of the Lyapunov function along solutions of a dynamical system is upper-bounded by the maximum of the Clarke generalized derivative. We use the notation from both [58] and [59], and [59] provides the proof.

Let the mode function $I(x) : \mathbb{R}^N \to I_{\theta_V}$ be given by [59]

$$I(x) = \{i \in I_{\theta_V} : x \in X_i\}, \tag{38}$$

where $X_i \in \mathcal{P}_{\theta_V}$. Clearly, this function enumerates which cells $x$ belongs to. This function is single-valued on the interior of cells in $\mathcal{P}_{\theta_V}$ and set-valued on boundaries between cells.

The Clarke generalized gradient $\partial V(x)$ is given [58] by

$$\partial V(x) = \overline{\operatorname{conv}} \left( \lim_{k \to \infty} \nabla V(x_k) : x_k \to x, x_k \notin \mathcal{N}_V \right),$$

where $\mathcal{N}_V \subset \mathbb{R}^n$ is a set of measure zero where $\nabla V(x)$ is not defined. For the piecewise affine function of the form (13), the function $\partial V(x)$ reduces to

$$\partial V(x) = \operatorname{conv} \left( \{p_i\}_{i \in I(x)} \right),$$

where $p_i$ for $X_i$ is given by (18).

The Clarke generalized derivative $\dot{V}_F$ for a differential inclusion $\dot{x} \in F(x)$ is given [58] by

$$\dot{V}_F(x) = \left\{ p^T f : p \in \partial V(x), \; f \in F(x) \right\}.$$

When $F(x)$ is a singleton for all $x \in \mathbb{R}^n$, say $F(x) = f(x)$, as is in our case, this condition reduces to

$$\dot{V}_F(x) = \left\{p^T f(x) \colon p \in \partial V(x)\right\}. \tag{39}$$

If the conditions (33)-(37) hold, then by Lemmas 1 and 2, we may conclude that $p_i^T(A_i^T x + a_i) < 0$ for all $x \neq 0$, $x \in X_i \in \mathcal{P}_{\theta_V}$, for each $i \in I_{\theta_V}$. This enumeration over all $i \in I_{\theta_V}$ ensures that this condition is checked at $x$ for each $i \in I(x)$. Therefore, we may conclude that $p^T f < 0$ for all $p \in \partial V(x)$, at each $x \neq 0$, where $f$ is the single-valued dynamics at $x$. From (39) we can conclude that $\dot{V}_F(x) < 0$ at each $x \neq 0$. In turn, from Proposition 3.2 in [59], we may conclude that $\dot{V}(t) < 0$ almost everywhere along solutions of the dynamical system, so that the origin of (2) is (locally) asymptotically stable. ■

## C. Summary

To summarize, we have derived exact quantifier-free conditions on the parameters $\theta_V$ of a candidate Lyapunov function $V(x)$ that when satisfied allows us to conclude that it is a valid Lyapunov function certifying that the origin of a dynamical system of the form (2) is asymptotically stable. In the next sections, we use these conditions to propose new algorithms to search for a Lyapunov function using sequential convex optimization, and to synthesize controllers for controlled systems of the form (1).

## VII. Algorithms For Stability Analysis

This section proposes an algorithm for finding Lyapunov functions of the form (13) that verify stability properties of the origin of dynamical systems of the form (2). We exploit the idea that the role of the hidden neurons parametrized by $(H_V, b_V)$ defines a partition, while the output layer weights $w_V$ defines a function over that partition. This idea leads to an iterative approach based on alternating between a) choosing the weights $w_V$ and $c_V$ of the output layer of the ReLU NN Lyapunov function for a given set of neurons using numerical optimization, and b) adding new neurons based on the results of this optimization. We use optimization problem $\mathrm{Opt}_{\theta_V}$ in Section VII-A to choose $w_V$ and $c_V$, wherein $H_V$, $b_V$ are fixed. If the optimal value is non-zero, we use the solution to insert new neurons into the Lyapunov ReLU neural network. In effect, we reformulate the non-convex optimization problem typically solved using gradient descent as an alternation between convex optimization and neural architecture search.

## A. 'Learning' Outer-Layer Weights

The conditions in Theorem 4 allow us to search for a Lyapunov function $V(x)$ given $(H_V, b_V)$ using linear programming. If this linear program is feasible, we verify the asymptotic stability of the origin. What should we do when it is infeasible? We solve this problem by introducing slack variables for some of the constraints, and changing the objective function to the sum of the norms of these slack variables.

For each $i \in I_{\theta_V}$, we introduce slack variables $s_i$ for constraints (33) and (35), and slack variables $t_i$ for (36). We then define the objective function $J_{H_V,b_V}(w_V, c_V)$ given by

$$J_{H_V,b_V}(w_V, c_V) = \sum_{i \in I_{H_V,b_V}} \|s_i\|_2^2 + \|t_i\|_2^2. \tag{40}$$

The optimization problem $\mathrm{Opt}_{\theta_V}$ that implements a search for $w_V$ given fixed $(H_V, b_V)$ is

$$\begin{aligned}
&\min_{w_V, c_V, \mu_i, \nu_i, s_i, t_i} && J_{H_V,b_V}(w_V, c_V) && (41)\\
&\text{s.t.} && w_V \max.(b_V, 0) + c_V = 0, && (42)\\
& && H_V^T S_i^{\theta_V} \mu_i - p_i = 0, \quad \forall i \in I_{\theta_V}^{lin}, && (43)\\
& && \mu_i \succeq \epsilon_1 \mathbf{1}, \quad \forall i \in I_{\theta_V}^{lin}, && (44)\\
& && H_V^T S_i^{\theta_V} \mu_i - p_i = 0, \quad \forall i \in I_{\theta_V}^{aff}, && (45)\\
& && b_V^T S_i^{\theta_V} \mu_i - q_i + \epsilon_2 \leq 0, \quad \forall i \in I_{\theta_V}^{aff}, && (46)\\
& && \mu_i \succeq 0, \quad \forall i \in I_{\theta_V}^{aff}, && (47)\\
& && H_V^T S_i^{\theta_V} \nu_i + A_i^T p_i = s_i, \quad \forall i \in I_{\theta_V}^{lin}, && (48)\\
& && \nu_i \succeq \epsilon_2 \mathbf{1}, \quad \forall i \in I_{\theta_V}^{lin}, && (49)\\
& && H_V^T S_i^{\theta_V} \nu_i + A_i^T p_i = s_i, \quad \forall i \in I_{\theta_V}^{aff}, && (50)\\
& && \epsilon_2 + b_V^T S_i^{\theta_V} \nu_i + a_i^T p_i \leq t_i, \quad \forall i \in I_{\theta_V}^{aff} && (51)\\
& && \nu_i \succeq 0, \quad \forall i \in I_{\theta_V}^{aff}, && (52)
\end{aligned}$$

where $\epsilon_1 > 0$ and $\epsilon_2 > 0$. The variables $p_i$, $q_i$, $A_i$, and $a_i$ are defined in (18)-(21). The optimization problem $\mathrm{Opt}_{\theta_V}$ in (41)-(52) contains several variables, of which $w_V$ and $c_V$ are most important, and the rest are related to establishing properties of the function $V(x)$. We state the following result:

**Lemma 5.** *The optimization problem* $\mathrm{Opt}_{\theta_V}$ *in* (41)-(52) *is always feasible.*

*Proof.* This result is by construction. By including $n$ neurons in $H_V, b_V$ that have no bias terms and correspond to independent vectors in $\mathbb{R}^n$, along with their reflections, we ensure that there exists a choice for $w_V$ that makes $V(x)$ positive definite. Therefore constraints (42)-(47) will always be feasible by themselves. The remaining constraints are always feasible for any value of $w_V$ due to the use of slack-like variables $s_i$ and $t_i$. Therefore $\mathrm{Opt}_{\theta_V}$ is always feasible. ■

While $\mathrm{Opt}_{\theta_V}$ is always feasible, we desire the optimal value to be zero, due to Theorem 4. The next section describes how we continue searching for a valid ReLU NN Lyapunov function when the optimal value of $\mathrm{Opt}_{\theta_V}$ is non-zero.

## B. Adding Neurons

In this work, we add neurons $(H^{new}, b^{new})$ such that they define hyperplanes that split cells $X_i \in \mathcal{P}_{\theta_V}$ for which slack variables $s_i$ or $t_i$ in the solution to $\mathrm{Opt}_{\theta_V}$ are non-zero. This process is based on the idea that we need to refine such cells, as presented in [17], [42], [44]. The approach here is simpler than in [42], [44].

Consider two index sets $I_s$ and $I_t$, where

$$\begin{aligned}
I_s &= \{i \in I_{\theta_V} \colon s_i \neq 0\}, \text{ and}\\
I_t &= \{i \in I_{\theta_V} \colon t_i \neq 0\}.
\end{aligned}$$

At least one of these sets is non-empty when $\text{Opt}_{\theta_V}$ has non-zero optimal value. These index sets lead to the following rules for refining cells:

1) If $I_s$ is not empty, split all cells in $I_s$.
2) Otherwise, split all cells in $I_t$.

If $i \in I_s$, we simply choose the neurons to be

$$(H^{new},\ b^{new}) = \left(w_V S_i^{\theta_V} H_V, w_V S_i^{\theta_V} b_V\right) \tag{53}$$

and its reflection. The reason is that if $i \in I_s$, then the hyperplane $H^{new}x + b^{new} = 0$ may divide $X_i$ into two regions where $\dot{V} < 0$ for one and $\dot{V} > 0$ for the other. Adding these two neurons may allow the algorithm to modify the Lyapunov function on exactly one of these regions.

If $i \in I_t$, we simply choose the neurons to be

$$(H^{new},\ b^{new}) = \left(\left(W_x S_i^{\theta_V} b_V\right)^T, 0\right) \tag{54}$$

and its reflection. The reason is that $i \in I_t$ implies $p_i^T a_i > 0$, which needs to be reduced. By adding $(a_i^T, 0)$ and $(-a_i^T, 0)$ as neurons, the algorithm may be able to modify $p_i$ for this cell (or subsets of it) to $p_i' = p_i - \epsilon a_i$. If this modification happens, $(p_i')^T a_i < p_i^T a_i$, achieving the desired reduction.

These choices are well-motivated but do not provide any *a priori* guaranteed improvements. Each added neuron is designed to allow modifications of $V(x)$ for a specific cell, but they can also cause changes to $V(x)$ in other cells. We are currently unable to analyze this phenomenon. Therefore, we cannot even guarantee that the added neurons will change the candidate Lyapunov function. Empirically, refinements appear to lead to improvements, as also seen in our examples and also in [42], [44].

## C. Our Search Algorithm

Algorithm 1 describes the algorithm resulting from these choices. We can show the following properties.

**Proposition 6.** *Algorithm 1 is sound.*

*Proof.* The algorithm terminates only when $\text{Opt}_{\theta_V}$ has an optimal value of zero. By Theorem 4, the candidate Lyapunov function $V(x)$ is therefore a valid Lyapunov function certifying asymptotic stability of the origin. ∎

Algorithm 1 is non-terminating. If we attempt to use Algorithm 1 to find a Lyapunov function for the origin of the harmonic oscillator, it will not terminate, and instead will indefinitely insert neurons after every iterate of optimization.

For the case where $\epsilon_2 = 0$, corresponding to certifying only stability of the origin, Algorithm 1 will fail to find one, implying that it is not complete.

*Computational Complexity.:* The optimization problem $\text{Opt}_{\theta_V}$ is a quadratic program. Let there be $m$ neurons in $\theta_V$. Then there are a total of $m+n+|\mathcal{C}_m|(2m+n)$ variables and $1+2|\mathcal{C}_m|(m+n)$ constraints, where $n$ is the dimension of the state and $|\mathcal{C}_m|$ are the number of cells in the arrangement defined by $\theta_x$.

The parameters $w_V$ and $c_V$ introduce $m+n$ variables. Each cell $X_i \in \mathcal{P}_{\theta_V}$ introduces $2m+n$ variables and $2m+2n$ constraints, with one more variable $t_i$ and two more constraints if $i \in I_{\theta_V}^{aff}$.

Despite the formulation as a convex optimization problem, solving it is not fast for large $m$. In the worst case, $|\mathcal{C}_m|$ can have $\zeta(m,n)$ cells (see (24)) which is super-polynomial [55].

**Algorithm 1** Verifying Stability using Lyapunov ReLU NNs

```
Require: $\text{NN}_{\text{ReLU}}(x,\theta_x)$, $\epsilon_1 > 0$, $\epsilon_2 \geq 0$
Ensure: ReLU NN Lyapunov function $V(x)$ that verifies the origin is (asymptotically) stable.
  $\theta_V \leftarrow \mathbf{0}_n^T\theta_x$ {Initialize network with neurons of $\theta_x$}
  $J_{H_V,b_V}(w_V,c_V) \leftarrow \infty$
  while $J_{H_V,b_V}(w_V,c_V) \neq 0$ do
    Solve $\text{Opt}_{\theta_V}$.
    $I_s \leftarrow \{i \in I_{\theta_V} : s_i \neq 0\}$
    $I_t \leftarrow \{i \in I_{\theta_V} : t_i \neq 0\}$
    for $i \in I_s$ do
      Compute $(H^{new}, b^{new})$, add to $(H_V, b_V)$ (see Section VII-B)
    end for
    if $I_s = \emptyset$ then
      for $i \in I_t$ do
        Compute $(H^{new}, b^{new})$, add to $(H_V, b_V)$ (see Section VII-B)
      end for
    end if
  end while
  return $V(x) = w_V\sigma_+.(H_V x + b_V)$.
```

## D. Extensions To Other Closed-Loop Properties

The discussion so far has focused on asymptotic stability. The extension to properties such as stability, exponential stability, and ultimate boundedness are straightforward.

*1) Stability:* We simply choose $\epsilon_2 = 0$.

*2) Exponential Stability:* We may show exponential stability by requiring that $\dot{V} \leq -\alpha V$ for some $\alpha > 0$. This constraint implies that for each cell $X_i \in I_{H_V,b_V}$, we need

$$p^T(Ax+a) < -\alpha p^T x, \text{ or } \left(p^T A + \alpha p^T\right) + p^T a < 0.$$

The constraints in our optimization-based algorithms barely change, with the linear dynamics matrix being modified from $A$ to $A + \alpha I$. We must either choose $\alpha$ through some side information, or use some procedure to search for it. We may use methods inspired by bisection algorithms, however, we do not pursue this further here.

*3) Ultimate Boundedness:* In some cases, we may need to show ultimate boundedness to some set $X_0$ containing the origin, instead of asymptotic stability. For example, when models are learned from data, the vector field at the origin may be non-zero, even if small. Instead of adding a correction term, which would be added to the vector field at all states, we may show the ultimate boundedness of the learned dynamics.

If the set $X_0$ is characterized by the constraint $Ex + e \geq 0$, then we can add $(E, e)$ to the neurons $(H_V, b_V)$. The enumeration algorithm in [55] will enumerate all subsets of $X_0$, and we can ignore these sets when constructingthe Lyapunov decrease conditions (48)-(49).

*4) Convex Lyapunov Functions:* If we constrain the outer weights $w_V$ of the Lyapunov ReLU neural network to satisfy $w_V \succeq 0$, then the Lyapunov function will be convex.

## VIII. Algorithms For Controller Synthesis

Recall that we wish to find a controller of the form $u(x) = \text{NN}_{\text{ReLU}}(x, \theta_u)$, where $\theta_u = (W_u, H_u, b_u, c_u)$, for a system of the form (1). The resulting closed loop will be

$$\dot{x} = \text{NN}_{\text{ReLU}}(x, \theta_x), \text{ where}$$
$$\theta_x = \theta_x' \oplus B\theta_u.$$

We again choose candidate $V(x) = \text{NN}_{\text{ReLU}}(x, \theta_x)$.

With these choices, we may represent the dynamics terms $A_i$ and $a_i$ in the stability condition (22) for each cell in $\theta_V$ as

$$A_i = (W_x + BW_u)\, D_i^{\theta_V} H_V, \text{ and} \tag{55}$$
$$a_i = (W_x + BW_u)\, D_i^{\theta_V} b_V. \tag{56}$$

For synthesis, we minimize the same objective as in (40). We may then derive stability conditions for verifying asymptotic stability of the controlled closed-loop that are identical to those in Theorem 4, except the affine dynamics are now given by (55) and (56) instead of (20) and (21). As a result, we will obtain a nearly identical optimization problem $\text{Opt}_{\text{synth}}(H_V, b_V)$ to $\text{Opt}_{\theta_V}$, except that the constraints are now bilinear in the optimization variables, with additional variable $W_u$. This bilinearity arises because the terms $A_i$ and $a_i$ in (55) and (56) depend on the optimization variables.

We use alternate convex search (ACS) [61] to solve this optimization problem with bilinear constraints. Each convex problem arises by keeping either $w_V$ or $W_u$ fixed, at the value of the solution from the previously solved optimization problem. Therefore, we obtain a nested iterative algorithm, where the inner optimization finds local minima for the bilinear problem $\text{Opt}_{\text{synth}}(H_V, b_V)$, and the outer optimization enables the insertion of nodes so that the local optima may have value zero. This algorithm is nearly identical to Algorithm 1, except that we solve $\text{Opt}_{\text{synth}}(H_V, b_V)$ using ACS, instead of $\text{Opt}_{\theta_V}$ using quadratic programming.

### A. Constrained Controller Synthesis

The proposed framework makes it easy to include polytopic constraints on the controller $u = \text{NN}_{\text{ReLU}}(x, \theta_u)$. Since these constraints are affine in $u$, they are in turn piecewise affine in $x$. If we use the neurons in the candidate Lyapunov neural network to define the controller, then we simply require that over each cell $X_i \in \mathcal{P}_{\theta_V}$, an additional set of affine inequalities in $x$ hold, which is a routine step in our approach.

## IX. Examples

We present four examples that demonstrate the performance of the search for a ReLU NN Lyapunov function proposed in Section VII. In addition, we present one example for the synthesis of a constrained controller as presented in Section VIII. We use the `Mosek` optimization package and `Julia v1.6` to implement all computations, using a computer with a 2.6 GHz processor and 16 GB RAM. The values of $\epsilon_1$ and $\epsilon_2$ are set to 1 and 0.001 respectively. We use a tolerance of $10^{-8}$ when checking if a number is non-zero. Table I summarizes the performance of Algorithm 1 and its extension to controller synthesis on the examples below.

| Example | Enumeration time (sec) | Computation time (sec) | Neurons | Regions | Verified property |
|---|---|---|---|---|---|
| 1 | 27.0 | 75.23 | 347 | 356 | LAS |
| 2 | 3.85 | 17.79 | 134 | 492 | LAS |
| 3 | 2.15 | 29.87 | 64 | 926 | LAS |
| 4 | 0.93 | 1.07 | 26 | 160 | LAS |
| 5 | 11.35 | 208.10 | 216 | 904 | LS |

TABLE I: Summary of examples of applying the proposed methods. LS: Local stability; LAS: Local Asymptotic Stability.

**Example 1** (MPC)**.** We consider the control policy derived for a discrete-time linear dynamical system using Model Predictive Control, similar to [11], with dynamics

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u_t. \tag{57}$$

The MPC problem features the stage-wise quadratic cost $x_t^T x_t + u_t^2$, actuator constraints $|u| < 4$, and state constraint $\|x\|_\infty < 5$. We use the `MPT3` toolbox in `Matlab` to obtain an explicit controller. We verify the stability of this explicit MPC controller when applied to the system

$$\dot{x} = \begin{bmatrix} 0 & 100 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 75 \\ 50 \end{bmatrix} u, \tag{58}$$

which produces the discrete-time dynamical system in (57) under a discretization time of 0.01 seconds. Figure 1 depicts the level sets of the Lyapunov function found by Algorithm 1.

While we find a Lyapunov function in Example 1, we encountered examples related to explicit MPC where the number of regions and neurons involved become too large to manage in our implementation.

**Example 2** (Wheeled vehicle path following)**.** The wheeled vehicle path following is another typical nonlinear example in the literature. We used the kinematic model as follows [5]:

$$\dot{d}_e = \nu sin(\theta_e), \tag{59}$$
$$\dot{\theta}_e = \omega - \frac{\nu\kappa(s)cos(\theta_e)}{1 - d_e\kappa(s)}.$$

In (59) state variables are $\theta_e$, the angle error, and $d_e$, the distance error, and the control is $\omega$. We used the NN controller designed in [5], and assumed that the target path is a unit circle, $\kappa(s) = 1$. We identified the closed-loop dynamic within $\|x\|_\infty \leq 0.8$ with 50 neurons using single-hidden layer ReLU. The Lyapunov function was obtained by applying the verification method to the identified dynamics. In verification, we exclude a neighborhood of the origin corresponding to the constraint $\|x\|_\infty \leq 0.005$. Figure 2 depicts the region of attraction and the vector fields. The results demonstrate that the region of attraction obtained by our proposed method is less conservative than that of the NN Lyapunov function in [5].

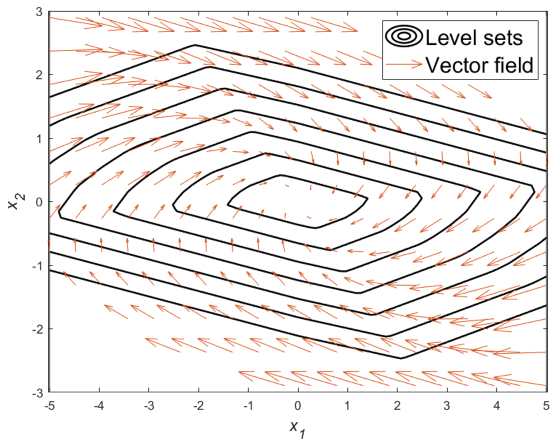

Fig. 1: Depiction of the closed-loop dynamics (arrows) and the level sets (curves) of the Lyapunov function verifying asymptotic stability of the origin for Example 1.

**Example 3** (Saturated PID Control)**.** We consider a spring-mass system driven by a PID control under saturation. This example is inspired by a similar one in [62]. The dynamics of the position $x$ with input $F$ is

$$\begin{aligned} m\ddot{x} + kx &= F, \text{ where} \\ F &= \operatorname{sat}\left(-K_p x - K_d \dot{x} - K_i y, 4\right), \\ \dot{y} &= x, \text{ and} \\ \operatorname{sat}(x,\alpha) &= \begin{cases} \alpha & \text{if } x \geq \alpha, \\ -\alpha & \text{if } x \leq -\alpha, \text{ or} \\ x & \text{otherwise .} \end{cases} \end{aligned}$$

The parameter values are $m = 0.1$ Kg, $k = 2$ N/m, $K_p = 24$ N/m, $K_d = 3.2$ Ns/m and $K_i = 44$ N s$^2$/m.

Note that we can implement $\operatorname{sat}(Hx, \alpha)$ as $\sigma_+(Hx) - \sigma_+(-Hx) - \sigma_+(Hx - \alpha) - \sigma_+(-Hx - \alpha)$, which is a ReLU NN with $4$ neurons.

**Example 4** (4D Problem)**.** We consider a version of a 4D problem from [62]. Our version has also two dynamics modes but is continuous. We also allow the boundary to be an affine subspace that does not pass through the origin. The method from [62] is inapplicable to it.

$$\begin{aligned} \dot{x} &= Ax - W\sigma+.\left(\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} x - 0.1\right), \text{ where} \\ W &= \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T, \text{ and} \\ A &= \begin{bmatrix} -3.01 & 1.0 & 1.0 & 1.0 \\ 1.0 & -3.01 & 1.0 & 1.0 \\ 1.0 & 1.0 & -3.01 & 1.0 \\ 1.0 & 1.0 & 1.0 & -3.01 \end{bmatrix}. \end{aligned}$$

**Example 5** (Inverted Pendulum)**.** We approximate the uncontrolled dynamics of a simple pendulum using a single-hidden-layer ReLU neural network with 20 neurons. Moreover, We employed the constrained ReLU controller, $|u| \leq 4$, as presented in VIII as a state feedback controller. The state-space dynamics of the pendulum are

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{c}{m}x_2 - gl^2\sin(x_1)\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{ml^2}\end{bmatrix} u$$

where $m = 0.15$ kg, $l = 0.5$ m, c $= 0.1 Ns/rad$, and $g = 9.81$ m/s$^2$ [10]. We use the ADAM optimization algorithm provided by the `Optim` package in `Julia`, with a learning rate of $1e^{-4}$, on $10000$ samples with batch size 3000. We do the control synthesis and stability analysis on the region corresponding to $\|x\|_\infty \leq 4$, excluding a neighborhood of the origin corresponding to the constraint $\|x\|_\infty \leq 0.1$. Figure 3 depicts the region of attraction found by Algorithm 1. In addition, we compared this region to LQR [9], NN method [5]. The region of attraction is less conservative than the LQR and NN; however, this method is not designed to maximize it.

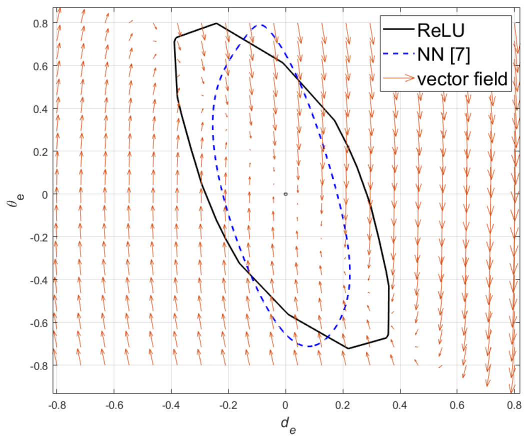

Fig. 2: Depiction of the closed-loop dynamics (arrows) and the region of attraction (curves) of NN [5] and the presented verification method. The Lyapunov function verifying asymptotic stability of the origin for Example 2.

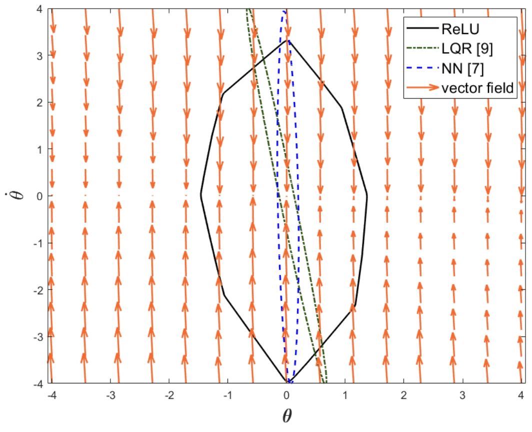

Fig. 3: Depiction of the level sets of the Lyapunov function verifying the stability of the origin and its comparison with LQR [9], and NN [5] for Example 5.

## X. Discussion And Future Work

We have presented examples to demonstrate successful applications of our automated approach to stability verification and controller synthesis. We see an advantage to our methods when the piecewise affine dynamics (closed-loop or otherwise) have more than a few modes. However, the neurons of the Lyapunov function create additional pieces in the partition we must enumerate. We can handle hundred of regions, but not thousands. Below, we discuss the challenges that exist in making this algorithm useful for a wider set of problems.

*Limitations:* The proposed solution relies on the piecewise affine nature of the dynamics. For some dynamical systems, a good approximation of the vector field using a ReLU NN may require a large number of neurons, making our optimization-based search and synthesis method intractable.

Our method to add neurons is heuristic; the effect on future iterations is unknown. Adding neurons increases the expressiveness of the network, which should aid the search for a valid Lyapunov function, but this claim is not formal.

The bilinear nature of the optimizations used for controller synthesis implies that this procedure is less reliable than the analysis algorithm. However, this bilinearity is largely unavoidable when simultaneously searching for controllers and Lyapunov-based certificates.

*Future Work:* This work presents a few clear avenues for future work. First, the characterization of dynamical systems and closed-loop properties for which a piecewise affine Lyapunov function must exist will guide the application of the proposed methods [39]. Second, our algorithm may benefit from allowing other methods to change the hidden neurons beyond only adding neurons heuristically. Third, we formulate stability conditions based on a halfspace representation of the cells of the partitions associated with the dynamics and Lyapunov function ReLU NNs. Finally, the algorithm focuses on continuous dynamical systems, which polynomial methods may also handle. The piecewise nature suggests that these methods will provide greater value for discontinuous dynamics or controllers, for which analysis and synthesis are difficult [18]. Automating the more involved technical issues is likely feasible [42], but possibly challenging [57]–[59].

## XI. Conclusion

This paper introduces a computational framework to verify and search for single-layer ReLU NN Lyapunov functions that verify properties of piecewise affine dynamical systems potentially represented as single-layer ReLU NN dynamical systems. We also extend this search procedure to enable controller synthesis. The design of our algorithms makes them sound by construction. Furthermore, we avoid the need to train networks using gradient descent. Inspired by refinement-based approaches, we instead increase the number of neurons in the network, thereby increasing its complexity.

While the algorithms are sound, several features are potentially computationally expensive, and the performance of the algorithm for searching for Lyapunov functions is not well characterized. Despite these challenges, this work points to promising automated algorithms for analyzing dynamics models learned from data and even synthesizing controllers for them. We anticipate that the potential for optimized implementations, and the relentless advance of computational power, to be factors in favor of such methods.

## References

[1] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[2] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999–1013, 1993.

[3] A. Toriello and J. P. Vielma, "Fitting piecewise linear continuous functions," *European Journal of Operational Research*, vol. 219, no. 1, pp. 86–95, 2012.

[4] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[5] R. Zhou, T. Quartz, H. D. Sterck, and J. Liu, "Neural lyapunov control of unknown nonlinear systems with stability guarantees," in *Advances in Neural Information Processing Systems*, 2022. [Online]. Available: https://openreview.net/forum?id=QvlcRh8hd8X

[6] J.-N. Lin and R. Unbehauen, "Canonical piecewise-linear approximations," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, no. 8, pp. 697–699, 1992.

[7] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[8] H. Ravanbakhsh and S. Sankaranarayanan, "Learning lyapunov (potential) functions from counterexamples and demonstrations," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[9] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dé Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[10] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 466–476.

[11] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning lyapunov functions for piecewise affine systems with neural network controllers," *arXiv preprint arXiv:2008.06546*, 2020.

[12] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.

[13] S. Boyd, V. Balakrishnan, E. Feron, and L. ElGhaoui, "Control system analysis and synthesis via linear matrix inequalities," in *1993 American Control Conference*, June 1993, pp. 2147–2154.

[14] S. Prajna and A. Papachristodoulou, "Analysis of switched and hybrid systems - beyond piecewise quadratic methods," in *Proceedings of the American Control Conference.*, vol. 4, June 2003, pp. 2779–2784.

[15] S. Prajna, P. A. Parrilo, and A. Rantzer, "Nonlinear control synthesis by convex optimization," *IEEE Transactions on Automatic Control*, vol. 49, no. 2, pp. 310–314, Feb 2004.

[16] M. Johansson and A. Rantzer, "Computation of piecewise quadratic Lyapunov functions for hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 555–559, Apr 1998.

[17] M. Johansson, "Piecewise linear control systems," Ph.D. dissertation, Lund University, 1999.

[18] E. Treadway and R. B. Gillespie, "Vector field control methods for discretely variable passive robotic devices," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 375–389, 2021.

[19] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[20] M. Korda, "Stability and performance verification of dynamical systems controlled by neural networks: algorithms and complexity," *IEEE Control Systems Letters*, vol. 6, pp. 3265–3270, 2022.

[21] O. L. Mangasarian, *Nonlinear Programming*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1994.

[22] I. Polik and T. Terlaky, "A survey of the s-lemma," *SIAM Review*, vol. 49, no. 3, pp. 371–418, 2007.

[23] 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.*, vol. 3, 2002, pp. 3482–3487 vol.3.

[24] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, “Introducing sostools: a general purpose sum of squares programming solver,” in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 1, Dec 2002, pp. 741–746 vol.1.

[25] J. Anderson and A. Papachristodoulou, “Advances in computational lyapunov analysis using sum-of-squares programming,” *Discrete & Continuous Dynamical Systems - B*, vol. 20, p. 2361, 2015.

[26] R. Sepulchre and D. Aeyels, “Homogeneous lyapunov functions and necessary conditions for stabilization,” *Mathematics of Control, Signals and Systems*, vol. 9, no. 1, pp. 34–58, 1996.

[27] D. Liberzon, J. P. Hespanha, and A. Morse, “Stability of switched systems: a lie-algebraic condition,” *Systems & Control Letters*, vol. 37, no. 3, pp. 117 – 122, 1999.

[28] R. Goebel and R. Sanfelice, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[29] H. Lin and P. J. Antsaklis, “Stability and stabilizability of switched linear systems: A survey of recent results,” *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308–322, Feb 2009.

[30] J. Oehlerking, H. Burchardt, and O. Theel, “Fully automated stability verification for piecewise affine systems,” in *Hybrid Systems: Computation and Control*, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 741–745.

[31] R. Iervolino, F. Vasca, and L. Iannelli, “Cone-copositive piecewise quadratic Lyapunov functions for conewise linear systems,” *IEEE Tran. on Automatic Control*, vol. 60, no. 11, pp. 3077–3082, 2015.

[32] M. S. Branicky, “Multiple Lyapunov functions and other analysis tools for switched and hybrid systems,” *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, April 1998.

[33] M. A. Wicks, P. Peleties, and R. A. DeCarlo, “Construction of piecewise Lyapunov functions for stabilizing switched systems,” in *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 4, Dec 1994, pp. 3492–3497 vol.4.

[34] S. Pettersson and B. Lennartson, “Stabilization of hybrid systems using a min-projection strategy,” in *Proceedings of the 2001 American Control Conference*, vol. 1, 2001, pp. 223–228 vol.1.

[35] T. Hu, L. Ma, and Z. Lin, “Stabilization of switched systems via composite quadratic functions,” *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2571–2585, Dec 2008.

[36] P. Bolzern and W. Spinelli, “Quadratic stabilization of a switched affine system about a nonequilibrium point,” in *Proceedings of the 2004 American Control Conference*, vol. 5, June 2004, pp. 3890–3895.

[37] T. Soga and N. Otsuka, “Quadratic stabilizability for polytopic uncertain continuous-time switched linear systems by output feedback,” in *Proceedings of the American Control Conference*, June 2010, pp. 3920–3925.

[38] L. Hetel and E. Bernuau, “Local stabilization of switched affine systems,” *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 1158–1163, April 2015.

[39] F. Blanchini, “Nonquadratic Lyapunov functions for robust control,” *Automatica*, vol. 31, no. 3, pp. 451 – 461, 1995.

[40] H. A. Poonawala, N. Lauffer, and U. Topcu, “Training classifiers for feedback control with safety in mind,” *Automatica*, vol. 128, 2021.

[41] H. A. Poonawala, N. Lauffer, and U. Topcu, “Training classifiers for feedback control,” in *2019 American Control Conference (ACC)*, July 2019, pp. 4961–4967.

[42] H. A. Poonawala, “Stability analysis of conewise affine dynamical systems using conewise linear lyapunov functions,” *IEEE Control Systems Letters*, pp. 1–1, 2020.

[43] H. Dai, B. Landry, M. Pavone, and R. Tedrake, “Counter-example guided synthesis of neural network lyapunov functions for piecewise linear systems,” in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 1274–1281.

[44] H. A. Poonawala, “Stability Analysis Via Refinement Of Piece-wise Linear Lyapunov Functions,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 1442–1447.

[45] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, “Lyapunov-stable neural-network control,” 2021.

[46] C. Barrett and C. Tinelli, “Satisfiability modulo theories,” in *Handbook of Model Checking*. Springer, 2018, pp. 305–343.

[47] L. De Moura and N. Bjørner, “Z3: An efficient smt solver,” in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[48] R. Alur, S. Kannan, and S. La Torre, “Polyhedral flows in hybrid automata,” in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 1999, pp. 5–18.

[49] P. Prabhakar and M. G. Soto, “Counterexample guided abstraction refinement for stability analysis,” in *International Conference on Computer Aided Verification*, 2016, pp. 495–512.

[50] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, “Formal synthesis of lyapunov neural networks,” *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[51] D. Ahmed, A. Peruffo, and A. Abate, “Automated and sound synthesis of lyapunov functions with smt solvers,” in *Tools and Algorithms for the Construction and Analysis of Systems*, A. Biere and D. Parker, Eds. Cham: Springer International Publishing, 2020, pp. 97–114.

[52] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, “Learning lyapunov functions for hybrid systems,” in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[53] M. Farsi, Y. Li, Y. Yuan, and J. Liu, “A piecewise learning framework for control of unknown nonlinear systems with stability guarantees,” in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 830–843.

[54] T. Geyer, F. D. Torrisi, and M. Morari, “Efficient mode enumeration of compositional hybrid systems,” in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 216–232.

[55] M. Rada and M. Cérný, “A new algorithm for enumeration of cells of hyperplane arrangements and a comparison with avis and fukuda’s reverse search,” *SIAM Journal on Discrete Mathematics*, vol. 32, no. 1, pp. 455–473, 2018.

[56] D. Avis and K. Fukuda, “Reverse search for enumeration,” *Discrete applied mathematics*, vol. 65, no. 1-3, pp. 21–46, 1996.

[57] J. Cortes, “Discontinuous dynamical systems,” *IEEE Control Systems Magazine*, vol. 28, no. 3, pp. 36–73, June 2008.

[58] M. Della Rossa, A. Tanwani, and L. Zaccarian, “Smooth approximation of patchy Lyapunov functions for switched systems,” *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 364–369, 2019.

[59] R. Baier, L. Grüne, and S. F. Hafstein, “Linear programming based Lyapunov function computation for differential inclusions,” *Discrete & Continuous Dynamical Systems - B*, vol. 17, p. 33, 2012.

[60] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski, *Nonsmooth analysis and control theory*. Springer Science & Business Media, 2008, vol. 178.

[61] J. Gorski, F. Pfeuffer, and K. Klamroth, “Biconvex sets and optimization with biconvex functions: a survey and extensions,” *Math Meth Oper Res*, pp. 373–407, 2007.

[62] G. O. Berger and S. Sankaranarayanan, “Counterexample-guided computation of polyhedral lyapunov functions for hybrid systems,” *arXiv preprint arXiv:2206.11176*, 2022.

**Pouya Samanipour** Pouya Samanipour is a PhD student in the Department of Mechanical Engineering at the University of Kentucky. He received a Bachelor’s degree in Electrical Engineering from Imam Khomeini International University, Qazvin, Iran in 2011, and a Master’s degree in Electrical Engineering from Iran University of Science and Technology, Tehran, Iran in 2015. The current focus of his research toward a PhD degree is the stability analysis of PWA dynamics and neural networks.

**Hasan A. Poonawala** Hasan A. Poonawala is an Assistant Professor in the Department of Mechanical Engineering at the University of Kentucky. He holds a Master's degree in Mechanical Engineering from the University of Michigan (2009), and a Ph.D. in Electrical Engineering from the University of Texas at Dallas (2014). His research expertise spans mechatronics, vision-based motion control, and classifier-in-the-loop systems. His current research focuses on controlling robotic systems using high-dimensional sensor data, machine learning, and control theory.