Learning fixed-complexity polyhedral Lyapunov functions from counterexamples

Guillaume O. Berger and Sriram Sankaranarayanan

***Abstract*— We study the problem of synthesizing polyhedral Lyapunov functions for hybrid linear systems. Such functions are defined as convex piecewise linear functions, with a finite number of pieces. We first prove that deciding whether there exists an m-piece polyhedral Lyapunov function for a given hybrid linear system is NP-hard. We then present a counterexample-guided algorithm for solving this problem. The algorithm alternates between choosing a candidate polyhedral function based on a finite set of counterexamples and verifying whether the candidate satisfies the Lyapunov conditions. If the verification fails, we find a new counterexample that is added to our set. We prove that if the algorithm terminates, it discovers a valid Lyapunov function or concludes that no such Lyapunov function exists. However, our initial algorithm can be non-terminating. We modify our algorithm to provide a terminating version based on the so-called cutting-plane argument from nonsmooth optimization. We demonstrate our algorithm on numerical examples.**

I. Introduction

Stability analysis of dynamical systems is of great importance in control theory, since many controllers seek to stabilize a system to a reference state or trajectory [1], [2]. Lyapunov methods are commonly used to prove stability of dynamical systems using a positive-definite function, called a Lyapunov function, whose value strictly decreases along all trajectories of the systems, except at the equilibrium point.

In this paper, we focus on the stability analysis of hybrid linear systems. These are systems with multiple modes, each with a different continuous-time linear dynamics and state-dependent switching between the modes. These systems appear naturally in a wide range of applications, or as abstractions of more complex dynamical systems [3]. To study the stability of these systems, we rely on polyhedral functions, which are convex piecewise linear functions. This class of functions is interesting for Lyapunov analysis because the expressiveness of the function can be easily modulated by adjusting the number of linear pieces. Furthermore, for a large class of hybrid systems (including switched linear systems), if the system is stable, then a polyhedral Lyapunov function is guaranteed to exist for the system [4].

In this paper, we first establish that the problem of deciding whether there exists a polyhedral Lyapunov function with given number of linear pieces for a given hybrid linear system is NP-hard. Despite the important body of work on polyhedral Lyapunov functions, the study of the algorithmic complexity of the underlying problem when the number of linear pieces is fixed seems to be elusive. We fill this gap, thereby providing an insight on the complexity of algorithms meant to solve the problem with guarantees.

Next, we introduce a counterexample-guided algorithm to compute polyhedral Lyapunov functions for hybrid linear systems. Our approach alternates between choosing a candidate Lyapunov function based on a finite set of counterexamples and verifying whether the candidate satisfies the Lyapunov conditions. If the verification fails, we find a new counterexample that is added to our set. In effect, this new counterexample removes the current candidate (and many others) from further consideration. The learning–verification process is repeated until either a valid Lyapunov function is found or no such function is shown to exist for the system. The key challenges in this approach are twofold. First, there are multiple ways of removing candidates using a counterexample while keeping the set of remaining candidates convex. Our approach uses a tree-based search algorithm to explore these possibilities. Second, the algorithm as presented thus far would be non-terminating. We modify our approach to guarantee termination using the *cutting-plane argument*, wherein a careful choice of the candidate and the pruning of branches of the tree based on the inner radius of the feasible set of candidates are used to ensure termination and get upper bounds on the complexity of the algorithm. The output of the modified algorithm is either a polyhedral Lyapunov function for the system, or the conclusion that no "ϵ-robust" function exists for the system, where $\epsilon > 0$ is an input to the algorithm. In effect, we conclude that if a polyhedral Lyapunov function were to exist for the system, then perturbing its coefficients by more than ϵ in the L_2-norm will cause the function not to be a Lyapunov function.

Comparison with the literature. Blondel et al. provide a comprehensive introduction to the computational complexity of numerous decision problems related to control and stability of dynamical systems [5]. The problem of deciding stability of switched linear systems (a subclass of hybrid linear systems) is shown to be undecidable. The complexity of deciding whether a dynamical system admits a polyhedral Lyapunov function with a given number of linear pieces seems to not have been addressed so far. The same problem without the bound on the number of pieces is closely related to the problem of stability verification using general convex Lyapunov functions (see, e.g., [4]).

The problem of synthesizing Lyapunov functions for dynamical systems has received a lot of attention in the literature [1], [4]. Different classes of functions have been

G. Berger is funded in part by a fellowship from the Belgian American Educational Foundation (BAEF). This work was funded by the US National Science Foundation (NSF) under grant numbers CPS 1836900 and 1932189.
Both authors are affiliated with the University of Colorado Boulder. Emails: {guillaume.berger,srirams}@colorado.edu

considered for that purpose, resulting in various trade-offs between expressiveness and computational burden. Polynomial functions have proved useful for stability analysis [6], but they can be too conservative for some hybrid systems, as shown in Example 1. Piecewise polynomial Lyapunov functions allow to reduce the conservatism [7], at the cost of introducing hyperparameters and/or increasing the complexity. The computation of polyhedral Lyapunov functions has also received attention, for instance, using optimization-based [8]–[10] and set-theoretic methods [11], [12]. One drawback of these approaches is that they generally lack guarantees of convergence (to a valid Lyapunov function), or guarantees of termination and complexity, or involve many hyperparameters. By contrast, the only parameters in our approach are the number of linear pieces and the accuracy ϵ, and the algorithm always terminates and finds a polyhedral Lyapunov function, or ensures that no ϵ-robust such function exists for the system.

The idea of learning Lyapunov functions from data and verifying the result has been explored by many researchers, starting with [13], whose approach relies on random sampling of states, learning a candidate Lyapunov function from the samples and verifying the result. Our approach falls more precisely into the category of Counterexample-Guided Inductive Synthesis (CeGIS) techniques, which consist in iteratively adding counterexamples from the verification step. These techniques have been used in a wide range of control problems [14]–[20]. Particularly related to our work, [20] searches for neural-network Lyapunov functions with ReLU activation functions, using MILP for the verification; and [8] searches for polyhedral Lyapunov functions, using Linear Programming for the verification and updating the domain of the linear pieces from the counterexamples. However, both approaches lack guarantees of convergence and complexity. In a recent work [21], we also studied counterexample-guided approaches to compute polyhedral Lyapunov functions without a bound on the number of linear pieces.

Prabhakar and Soto present a CeGIS approach for verifying stability of polyhedral hybrid systems with piecewise constant differential inclusions by using counterexamples to drive a partitioning of the state-space [22]. The partitioning is used to compute a weighted graph whose cycles provide evidence of stability if the product of weights for each cycle is less than one. Failing this, we conclude potentially unstable behavior. Prabhakar and Soto's approach is distinct from our approach in two important ways: (a) it does not seek to compute a Lyapunov function but instead focuses on building a finite graph abstracting the behavior of the system and (b) they handle polyhedral inclusions, which are a different type of dynamics from those considered here.

Organization. Section II introduces the problem of interest and preliminary concepts related to Lyapunov analysis and polyhedral functions. In Section III, we describe the algorithm and its properties. Finally, in Section IV, we demonstrate the applicability on numerical examples.

Notation. $\mathbb{N}$ denotes the set of nonnegative integers. We let $\mathbb{N}_* = \mathbb{N} \setminus \{0\}$ and $\mathbb{R}^d_* = \mathbb{R}^d \setminus \{0\}$. For $n \in \mathbb{N}$, we let $[n] = \{1, \dots, n\}$.

II. Preliminaries and Problem Description

A. Hybrid linear systems

Hybrid linear systems consist of a finite set of *modes* Q, such that for each mode $q \in Q$, there is a closed polyhedral cone $\mathcal{H}_q \subseteq \mathbb{R}^d$ called the *domain*, and a *flow* matrix $A_q \in \mathbb{R}^{d\times d}$. Let $\mathcal{F} : \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ be the set-valued function defined by $\mathcal{F}(x) = \{A_q x : q \in Q,\, x \in \mathcal{H}_q\}$. For the purpose of our analysis, $\mathcal{F}$ describes completely the dynamics of the system. Therefore, in the following, we will refer to the system as *system* $\mathcal{F}$. A function $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ is a *trajectory* of system $\mathcal{F}$ if ξ is absolutely continuous and satisfies the differential inclusion $\xi'(t) \in \mathcal{F}(\xi(t))$ for almost all $t \in \mathbb{R}_{\geq 0}$ [23].

Remark 1: Note that the hybrid system considered above is a particular instance of hybrid linear systems, since there is no update of the continuous state at mode switches [3]. Extensions of our work to a more general class of hybrid systems will be considered in the future work.

B. Polyhedral Lyapunov functions

A *polyhedral function* $V : \mathbb{R}^d \to \mathbb{R}$ can be described as the pointwise maximum of a finite set of linear functions, i.e., $V(x) = \max_{i\in[m]} c_i^\top x$ where $m \in \mathbb{N}_{>0}$ and $\{c_i\}_{i=1}^m \subseteq \mathbb{R}^d$.

To be a Lyapunov function for system $\mathcal{F}$, a piecewise smooth function $V : \mathbb{R}^d \to \mathbb{R}$ must (i) be positive everywhere except at the origin, and (ii) strictly decrease along all trajectories of the system not starting at the origin [1]. In the case of polyhedral functions, a sufficient condition for being a Lyapunov function and thereby ensuring stability of the system, is as follows [24, Proposition 3]:

Proposition 1: A polyhedral function V with linear pieces $\{c_i\}_{i=1}^m$ is a Lyapunov function for system $\mathcal{F}$ if

(C1) for all $x \in \mathbb{R}^d_*$, there is $i \in [m]$ such that $c_i^\top x > 0$;
(C2) for all $x \in \mathbb{R}^d_*$, $v \in \mathcal{F}(x)$ and $i \in [m]$ with $c_i^\top x = V(x)$, it holds that $c_i^\top v < 0$.

In other words, a set a vectors $\{c_i\}_{i=1}^m$ represents a polyhedral Lyapunov function for system $\mathcal{F}$ if

$$(\forall\, x \in \mathbb{R}^d_*)\ (\forall\, i \in [m])\ \text{ either } (\exists\, j \in [m])\ c_i^\top x < c_j^\top x \\ \text{or } [\, c_i^\top x > 0 \ \text{ and } \ (\forall\, v \in \mathcal{F}(x))\ c_i^\top v < 0\,]. \tag{1}$$

The condition in (1) is difficult to enforce because (i) it involves an infinite set of constraints, and (ii) each constraint involves a disjunction of m clauses[1]. In the next section, we describe a tree-based counterexample-guided algorithm to tackle this problem. The idea is to (i) consider only a finite set of constraints, obtained from the counterexamples, and (ii) decompose the disjunction by introducing a branch in the tree for each clause.

We define the m-Lyap-Fun-Exists problem as that of checking the existence of a polyhedral function satisfying (1), given a hybrid linear system $\mathcal{F}$ and a number of pieces $m \geq 2$.

[1] $m-1$ clauses come from the "$\exists\, j$" and one clause comes from the condition between brackets "[. . .]".

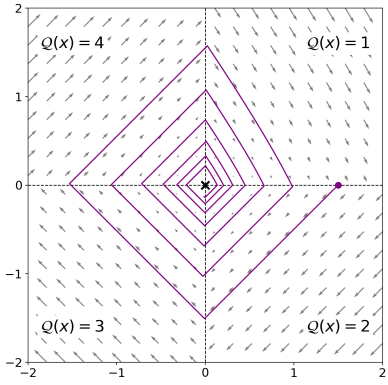

Fig. 1. Hybrid linear system described in Example 1. The vector field is represented by gray arrows and a sample trajectory starting from $x = [1.5, 0]^\top$ is represented in purple.

Theorem 2: The m-LYAP-FUN-EXISTS problem is NP-hard, even for $m = 2$.

Proof: See the extended version [25]. ■

To conclude this section, let us introduce an example of hybrid linear systems that we will use throughout the paper to illustrate the approach.

Example 1 (Running illustrative example): Consider the hybrid linear system with 4 modes, i.e., $Q = [4]$, domains corresponding to the 4 quadrants of the 2D plane, namely, $H_1 = \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$, $H_2 = \mathbb{R}_{\geq 0} \times \mathbb{R}_{\leq 0}$, $H_3 = \mathbb{R}_{\leq 0} \times \mathbb{R}_{\leq 0}$ and $H_4 = \mathbb{R}_{\leq 0} \times \mathbb{R}_{\geq 0}$, and flow matrices:

$$A_1 = \begin{bmatrix} \frac{1}{2} & 1 \\ -1 & \frac{-3}{2} \end{bmatrix}, \quad A_2 = A_4 = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix},$$
$$A_3 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} + \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

This system can be seen as a piecewise linear damped oscillator. The vector field and a trajectory starting from $x = [1.5, 0]^\top$ are represented in Fig. 1. The system does not admit a polynomial Lyapunov function (a proof is provided in the extended version [25]). Nevertheless, as we will see throughout the paper, we can compute a polyhedral Lyapunov function for the system, thereby ensuring that it is asymptotically stable.

III. COUNTEREXAMPLE-GUIDED SEARCH

We present a tree-based counterexample-guided algorithm to find a polyhedral function satisfying (1). Our approach maintains a *tree*, wherein each node is associated with a set of constraints. At each step, the algorithm picks a leaf of the tree and expands it as follows:

1) *Learning:* Solve a linear program to obtain a candidate polyhedral Lyapunov function compatible with the associated set of constraints Y.
2) *Verification:* The candidate solution is then fed to a verifier that checks whether it satisfies (1). If this fails, the verifier provides a pair $(x, i) \in \mathbb{R}^d_* \times [m]$, called a *counterexample*, for which (1) is not satisfied.
3) *Expansion:* If a counterexample is found, the algorithm creates m child nodes in the tree. Each child node is associated with a set of constraints obtained by adding to Y one of the clauses in (1) for the counterexample (x, i) (see below for details).

The sets of constraints associated to each node are sets of triples of the form $(x, i, j) \in \mathbb{R}^d_* \times [m] \times [m]$ wherein (x, i) are obtained from the counterexamples, and j is an index indicating which clause of (1) must be enforced at (x, i). More precisely, if $i \neq j$, we enforce the clause

$$c_i^\top x < c_j^\top x, \tag{2}$$

and if $i = j$, we enforce the clause

$$c_i^\top x > 0 \text{ and } (\forall\, v \in \mathcal{F}(x))\ c_i^\top v < 0. \tag{3}$$

Our approach systematically explores this tree and stops when one leaf of the tree yields a valid Lyapunov function. Under some modifications of the basic scheme above, we prove that the depth of this tree is bounded. This yields an upper bound on the running-time complexity of the algorithm.

The section is organized as follows: first, we describe the learning phase, then the verification phase, and finally the expansion phase which yields the overall recursive process.

A. Learning Phase

Given a set of constraints $Y \subseteq \mathbb{R}^d_* \times [m] \times [m]$, we aim to find vectors $(c_i)_{i=1}^m \subseteq \mathbb{R}^d$ compatible with Y according to conditions (2)–(3). Therefore, we solve the following feasibility problem:

$$\text{find} \quad c_i \in [-1, 1]^d, \quad i \in [m] \tag{4a}$$
$$\text{s.t.} \quad \begin{cases} (2) \text{ if } i \neq j \\ (3) \text{ if } i = j \end{cases}, \quad \forall\, (x, i, j) \in Y, . \tag{4b}$$

When Y is finite, (4) is a Linear Program and thus can be solved efficiently [26], [27]. We denote by $S(Y)$ the set of feasible solutions of (5). A set of vectors $(c_i)_{i=1}^m \subseteq \mathbb{R}^d$ is thus a candidate solution compatible with Y iff $(c_i)_{i=1}^m \in S(Y)$. Note that $S(Y)$ is a convex set.

B. Verification Phase

We verify whether a candidate solution $(c_i)_{i=1}^m$ computed in the learning phase provides a valid Lyapunov function for system $\mathcal{F}$, by checking whether it satisfies (C1) and (C2) in Proposition 1. If this is not the case, we generates a pair $(x, i) \in \mathbb{R}^d \times [m]$, called a *counterexample*, at which (1) is not satisfied.

First, we verify that (C1) holds for the candidate solution by searching for $x \in \mathbb{R}^d_*$ such that $V(x) \leq 0$. Without loss of generality, we can set x to be on the boundary of $[-1, 1]^d$. Therefore, we solve $2d$ Linear Programs, for each facet F of $[-1, 1]^d$:

$$\text{find} \quad x \in \mathbb{R}^d \tag{5a}$$
$$\text{s.t.} \quad c_i^\top x \leq 0, \quad \forall\, i \in [m], \tag{5b}$$
$$x \in F. \tag{5c}$$

If for all facet F, (5) is infeasible, then (C1) holds. However, if there is a facet F for which (5) has a feasible solution x,

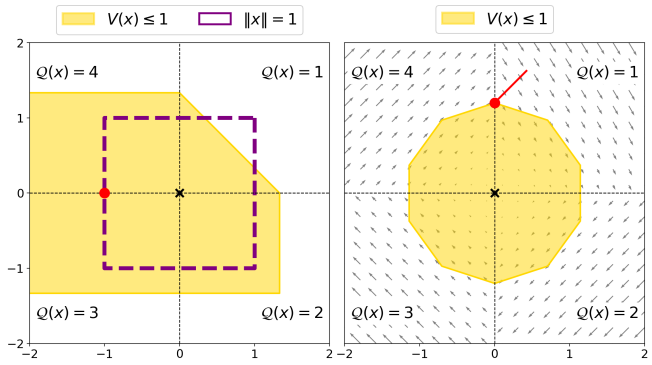

Fig. 2. *Left:* A point $x \in \mathbb{R}^d_*$ (in red) at which $V(x) \le 0$. *Right:* A point $x \in \mathbb{R}^d_*$ (in red) at which (C2) in Proposition 1 is not satisfied for the system described in Example 1.

then it holds that $x \neq 0$ and $V(x) \le 0$. We find index $i \in [m]$ with $c_i^\top x = V(x)$, yielding a counterexample (x, i).

If (C1) is shown to hold, we next verify that (C2) holds too. Therefore, we solve $m|Q|$ LPs, for each $i \in [m]$ and $q \in Q$,

$$\begin{aligned} \text{find} \quad & x \in \mathbb{R}^d && \text{(6a)} \\ \text{s.t.} \quad & c_i^\top x = 1 \ge c_j^\top x, \quad \forall\, j \in [m], && \text{(6b)} \\ & x \in \mathcal{H}_q, && \text{(6c)} \\ & c_i^\top A_q x \ge 0. && \text{(6d)} \end{aligned}$$

If for all i, q, (6) is infeasible, then (C2) holds. However, if there are i, q for which (6) has a feasible solution x, then it holds that $x \neq 0$, $c_i^\top x = V(x)$ and $c_i^\top v \ge 0$ wherein $v = A_q x \in \mathcal{F}(x)$, so that (x, i) is a counterexample.

Example 2 (Running illustrative example): Consider the 4-piece polyhedral function V whose 1-sublevel set is represented in Fig. 2-left (in yellow). The point $x = [-1, 0]^\top$ (in red) provides a direction in which αx is in the 1-sublevel set of V for all $\alpha \ge 0$. This implies that $V(x) \le 0$, so that x is a feasible solution to (5).

Now, consider the system of Example 1 and the 10-piece polyhedral function V whose 1-sublevel set is represented in Fig. 2-right (in yellow). For $q = 4$, (6) has a feasible solution $x = [0, 1.2]^\top$ (red dot). The flow direction of the system from x is represented in Fig. 2-right (red line). We see that the flow direction points towards the exterior of the 1-sublevel set of V.

C. Expansion and Overall Algorithm

The algorithm organizes the set of constraints as a tree with the following properties: (a) each node u is associated with a set of constraints $Y(u)$ and thus with a convex set of candidates $S(u) := S(Y(u))$; (b) the root of the tree is associated with $Y(\text{root}) = \emptyset$; and (c) each node u has m children, $u_1, \ldots, u_m$, with associated sets of constraints $Y(u_j) = Y(u) \uplus \{(x, i, j)\}$, wherein (x, i) is the counterexample found during the verification of the candidate at node u. Each leaf of the tree is marked as UNEXPLORED or INFEASIBLE.

Each iteration of our algorithm is as follows:

Algorithm 1: Find Polyhedral Lyapunov Function.

Data: system $\mathcal{F}$, number of pieces m.
Result: $(c_i)_{i=1}^m$: polyhedral Lyapunov function coefficients.

1 Initial tree is set to a root with $Y(\text{root}) = \emptyset$.
2 **while** *Tree has* UNEXPLORED *leaves* **do**
3 Choose UNEXPLORED leaf u.
4 **if** $S(u)$ *is empty* **then** mark u as INFEASIBLE.
5 **else**
6 Choose $(c_i)_{i=1}^m \in S(u)$
7 Verify $(c_i)_{i=1}^m$ using (5) and (6).
8 **if** $(c_i)_{i=1}^m$ *is Lyapunov* **then** return $(c_i)_{i=1}^m$
9 **else**
10 Let (x, i) be the counterexample.
11 Add m children $u_1, \ldots, u_m$ to u.
12 **for** $j = 1, \ldots, m$ **do**
13 $Y(u_j) \leftarrow Y(u) \cup \{(x, i, j)\}$.
14 Mark u_j as UNEXPLORED.
15 Return "No Lyapunov Found"

1) Choose an UNEXPLORED leaf u.
2) Find a candidate $(c_i)_{i=1}^m \in S(u)$.
3) Verify the candidate using (5) and (6).
4) If we find a counterexample (x, i), we add m children, $u_1, \ldots, u_m$, to u wherein u_j is UNEXPLORED and has set of constraints $Y(u_j) = Y(u) \uplus \{(x, i, j)\}$.

Alg. 1 shows the pseudo-code. We will now establish some properties of Alg. 1. Let $(c_i)_{i=1}^m$ be a candidate chosen for some leaf u (Line 6) and (x, i) be a counterexample that was found for this candidate (Line 10). Consider $S(u_j)$ for some child u_j of u (Line 13).

Proposition 3: $S(u_j) \subseteq S(u)$ and $(c_i)_{i=1}^m \notin S(u_j)$.

Proof: The first part is straightforward since $Y(u_j) = Y(u) \uplus (x, i, j) \supseteq Y(u)$. To show that $(c_i)_{i=1}^m \notin S(u_j)$, note that by definition of (x, i), it holds that $c_i^\top x = V(x)$. Hence, for all $j \in [m] \setminus \{i\}$, (2) is not satisfied. Furthermore, (x, i) satisfies

$$c_i^\top x \le 0 \ \vee \ (\exists v \in \mathcal{F}(x))\ c_i^\top v \ge 0.$$

Thus, (3) is not satisfied. We thus conclude that $S(u_j)$ cannot contain the candidate $(c_i)_{i=1}^m$. ∎

Let us assume that the underlying system $\mathcal{F}$ has a polyhedral Lyapunov function, given by $(c_i^*)_{i=1}^m$, satisfying (1) and without loss of generality assume $\{c_i^*\}_{i=1}^m \subseteq [-1, 1]^d$.

Proposition 4: At every step of any execution of Alg. 1, there is an unexplored leaf v such that $(c_i^*)_{i=1}^m \in S(v)$.

Proof: See the extended version [25]. ∎

As a corollary, we get the soundness of the algorithm:

Corollary 5: If Alg. 1 returns vectors $(c_i)_{i=1}^m$, then the polyhedral function associated to $(c_i)_{i=1}^m$ is a Lyapunov function for system $\mathcal{F}$. However, if Alg. 1 returns "No Lyapunov Found", then system $\mathcal{F}$ does not admit an m-piece polyhedral Lyapunov function satisfying (1).

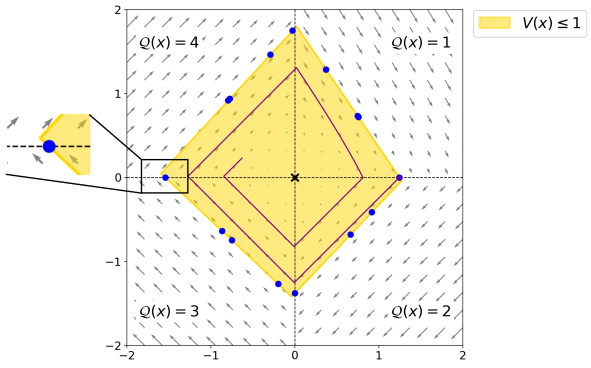

Fig. 3. 4-piece polyhedral Lyapunov function V for the system in Example 1. The blue dots are the states involved in the constraints leading to this polyhedral function. A sample trajectory starting from $x = [1.25, 0]^\top$ is represented in purple. Note the small "offset" (see zoomed-in inset): the domains of the linear pieces of V do not coincide with the domains of the hybrid system; this is necessary for condition (C2) in Proposition 1 to be satisfied at the boundary of these domains.

Proof: If Alg. 1 returns $(c_i)_{i=1}^m$, it means that $(c_i)_{i=1}^m$ has passed the verification phase (Line 8), thereby ensuring that it provides a Lyapunov function for system $\mathcal{F}$. If Alg. 1 terminates without finding a Lyapunov function, it means that all leaves are marked INFEASIBLE. By applying Proposition 4, we note that no Lyapunov function could have existed in the first place. ■

Example 3 (Running illustrative example): Consider the system of Example 1. A 4-piece polyhedral Lyapunov function for this system was computed using Alg. 1. Its 1-sublevel set is represented in Fig. 3 (in yellow). The states involved in the constraints $Y \in \mathbb{R}_*^d \times [m] \times [m]$ of the node that provided this function are represented by blue dots. We have also represented a trajectory of the system starting from $x = [1.25, 0]^\top$ (in purple). We observe that the trajectory does not leave the 1-sublevel set, as expected.

Note that there is no guarantee that Alg. 1 will terminate. To ensure termination, we modify the algorithm by leveraging a central result in convex nonsmooth optimization, known as the *cutting-plane* argument.

Therefore, assume that at Line 6 of Alg. 1, the candidate $(c_i)_{i=1}^m$ is chosen as the center of the Maximum Volume Ellipsoid (MVE) inscribed in $S(u)$. The MVE center of a polyhedron can be computed efficiently using Semidefinite Programming [27, Proposition 4.9.1]. Remember that the rationale of adding constraints from the counterexample (Line 13) is to exclude the candidate from further consideration at all descendant nodes. The choice of the candidate as the MVE center makes the exclusion stronger by guaranteeing a minimum rate of decrease of the volume of $S(u)$ when going from one node to any of its children. This follows from the cutting-plane argument:

Lemma 6 (Cutting-plane argument [28]): Let $\mathcal{S} \subseteq \mathbb{R}^r$ be a bounded convex set. Let x be the MVE center of $\mathcal{S}$. Let $\mathcal{T}$ be a convex set not containing x. It holds that $\mathrm{vol}(\mathcal{S} \cap \mathcal{T}) \leq (1 - \frac{1}{r})\mathrm{vol}(\mathcal{S})$.

Based on the above, consider the following modification of the algorithm. Fixed $\epsilon > 0$ and consider Alg. 1, except that in the learning phase, if $S(u)$ does not contain a ball of radius ϵ, then u is declared INFEASIBLE; otherwise, we choose the candidate $(c_i)_{i=1}^m$ as the MVE center of $S(Y)$. See Alg. 2 for a pseudo-code of the modified learning phase. It holds that the modified algorithm terminates in finite time.

Algorithm 2: Modify Alg. 1 Line 6 for termination.

1 **if** $S(u)$ *does not contain a ball of radius* ϵ **then** mark u as INFEASIBLE;
2 Choose $(c_i)_{i=1}^m$ as the MVE center of $S(u)$.

Theorem 7 (Termination and soundness): Alg. 2 always terminates. Moreover, if Alg. 2 returns vectors $(c_i)_{i=1}^m$, then the polyhedral function associated to $(c_i)_{i=1}^m$ is a Lyapunov function for system $\mathcal{F}$. If Alg. 2 returns "No Lyapunov Found", then system $\mathcal{F}$ does not admit an ϵ-robust polyhedral Lyapunov function, that is, a polyhedral function for which any (L_2) ϵ-perturbation of the linear pieces satisfies (1).

Proof: See the extended version [25]. ■

D. Complexity analysis

The dominant factor in the complexity of Alg. 2 is the depth D of the tree explored during the execution of the algorithm. From the proof of Theorem 7, it holds that

$$D \leq \frac{r \log(\epsilon)}{\log(1 - \frac{1}{r})}, \tag{7}$$

where $r = md$. Alg. 2 explores at most m^{D+1} nodes before termination. We deduce the following complexity bound:

Theorem 8: The complexity of Alg. 2 is

$$\mathcal{O}(m^{m^2 d^2 \log(1/\epsilon)}) \text{ FLOPs.}$$

Proof: Using the bound $-\log(1 - \frac{1}{r}) \geq \frac{1}{r}$, we get that $D \leq r^2 \log(\epsilon)$. The proof then follows from (7). ■

The complexity of Alg. 2 is exponential in d and m. The exponential dependence on d is to be expected from Theorem 2 which shows that m-LYAP-FUN-EXISTS is NP-hard, even with $m = 2$.

IV. NUMERICAL EXAMPLE

Consider the hybrid linear system with two modes, i.e., $Q = [2]$, domains $H_1 = \mathbb{R} \times \mathbb{R}_{\geq 0}$ and $H_2 = \mathbb{R} \times \mathbb{R}_{\leq 0}$, and flow matrices:

$$A_1 = \begin{bmatrix} \frac{-1}{2} & 1 \\ -1 & \frac{-1}{2} \end{bmatrix}, \quad A_2 = \begin{bmatrix} \frac{-3}{4} & 1 \\ -1 & \frac{-3}{4} \end{bmatrix}.$$

Fig. 4 shows the vector field and a sample trajectory of the system. Although a trivial quadratic Lyapunov function (e.g., $x \mapsto \|x\|^2$) exists for this system, finding a polyhedral one is challenging because many pieces are required to capture the rotational dynamics of the system.

Using the process described in Section III, we computed a 8-piece polyhedral Lyapunov function for the system. As an approximation of the MVE center, we used the Chebyshev center (center of the largest enclosed Euclidean ball), which

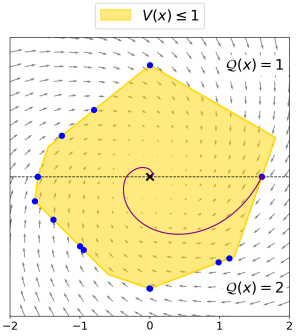

Fig. 4. Vector field (gray arrows), sample trajectory (purple line) and 8-piece polyhedral Lyapunov function (yellow) for the system described in Section IV. The blue dots are the states involved in the constraints leading to this polyhedral function.

can be computed using Linear Programming (more efficient and reliable than Semidefinite Programming). Although the theoretical guarantees on the termination of the process using the Chebyshev center are weaker than the ones with the MVE center (see Theorem 8), this provides a powerful heuristic in practice (see, e.g., [28, §4.4] and the references therein). The computation takes about 60 seconds.[2] The 1-sublevel set of the polyhedral Lyapunov function is represented in Fig. 4-right (in yellow).

V. Conclusions

In conclusion, we provide a NP-hardness result for finding fixed-complexity polyhedral Lyapunov function for hybrid linear systems. Furthermore, we describe a counterexample-driven approach based on switching between learning and verification followed by a tree of possible refutations. Our approach is modified to yield termination but at the cost of completeness. Our future work will focus on an efficient implementation of the procedure described in this paper and extensions to barrier and control Lyapunov function synthesis.

References

[1] H. K. Khalil, *Nonlinear systems*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.

[2] E. D. Sontag, *Mathematical control theory: deterministic finite dimensional systems*, 2nd ed. New York, NY: Springer, 1998.

[3] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid dynamical systems: modeling stability, and robustness*. Princeton, NJ: Princeton University Press, 2012.

[4] Z. Sun and S. S. Ge, *Stability theory of switched dynamical systems*. London: Springer, 2011.

[5] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.

[6] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, 2000.

[7] 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, 1998.

[8] A. Polański, "On absolute stability analysis by polyhedral Lyapunov functions," *Automatica*, vol. 36, no. 4, pp. 573–578, 2000.

[9] R. Ambrosino, M. Ariola, and F. Amato, "A convex condition for robust stability analysis via polyhedral Lyapunov functions," *SIAM Journal on Control and Optimization*, vol. 50, no. 1, pp. 490–506, 2012.

[10] D. Kousoulidis and F. Forni, "Polyhedral Lyapunov functions with fixed complexity," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 3293–3298.

[11] F. Blanchini and S. Miani, *Set-theoretic methods in control*, 2nd ed. Cham: Birkhäuser, 2015.

[12] N. Guglielmi, L. Laglia, and V. Protasov, "Polytope Lyapunov functions for stable and for stabilizable LSS," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 567–623, 2017.

[13] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669–2675, 2008.

[14] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Aréchiga, "Simulation-guided Lyapunov analysis for hybrid dynamical systems," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*. ACM, 2014, pp. 133–142.

[15] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," in *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*. ACM, 2019, pp. 3245–3254.

[16] H. Ravanbakhsh and S. Sankaranarayanan, "Learning control Lyapunov functions from counterexamples and demonstrations," *Autonomous Robots*, vol. 43, no. 2, pp. 275–307, 2019.

[17] H. A. Poonawala, "Stability analysis via refinement of piece-wise linear Lyapunov functions," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1442–1447.

[18] D. Ahmed, A. Peruffo, and A. Abate, "Automated and sound synthesis of Lyapunov functions with SMT solvers," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2020.*, ser. Lecture Notes in Computer Science, A. Biere and D. Parker, Eds., vol. 12078. Springer, 2020, pp. 97–114.

[19] 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, 2021.

[20] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Proceedings of Robotics: Science and Systems*, Virtual, 2021.

[21] G. O. Berger and S. Sankaranarayanan, "Counterexample-guided computation of polyhedral Lyapunov functions for hybrid systems," 2022, arXiv preprint arXiv:2206.11176.

[22] P. Prabhakar and M. G. Soto, "Counterexample guided abstraction refinement for stability analysis," in *Computer Aided Verification. CAV 2016.*, ser. Lecture Notes in Computer Science, S. Chaudhuri and A. Farzan, Eds., vol. 9779. Cham: Springer, 2016, pp. 495–512.

[23] J.-P. Aubin and A. Cellina, *Differential inclusions: set-valued maps and viability theory*. Berlin: Springer, 1984.

[24] 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.

[25] G. O. Berger and S. Sankaranarayanan, "Learning fixed-complexity polyhedral Lyapunov functions from counterexamples," 2022, arXiv preprint arXiv:2204.06693.

[26] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Philadelphia, PA: SIAM, 1994.

[27] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Philadelphia, PA: SIAM, 2001.

[28] S. Boyd and L. Vandenberghe, "Localization and cutting-plane methods," 2008, https://see.stanford.edu/materials/lsocoee364b/05-localization_methods_notes.pdf.

[2] On a laptop with processor Intel Core i7-7600u and 16 GB RAM running Windows. We use Gurobi™, under academic license, as linear optimization solver.