Polyhedral Control Lyapunov Functions for Switched Affine Systems

Sara Kamali
sara.kamali@colorado.edu
University of Colorado Boulder
USA

Guillaume Berger
guillaume.berger@uclouvain.be
UCLouvain
Belgium

Sriram Sankaranarayanan
srirams@colorado.edu
University of Colorado Boulder
USA

Abstract

We present a counterexample-guided approach for synthesizing convex piecewise affine control Lyapunov functions, obtained as the maximum over a finite number of affine functions, for stabilizing switched linear systems. Our approach considers systems whose dynamics are defined by a set of affine ODEs over different regions of the state-space. The goal is to synthesize a control feedback function that uses state-based switching by assigning a dynamical mode to each state from the set of available dynamics. This is achieved by synthesizing a piecewise affine control Lyapunov function that guarantees that for each state variable, the appropriate choice of a control input can cause an instantaneous decrease in the value of the Lyapunov function. Since piecewise affine functions are not smooth, we use a non-smooth analytic characterization of piecewise affine Lyapunov functions. The key contribution of our approach is a counterexample driven algorithm that alternates between *verification* that a given convex PWA function is a control Lyapunov function or generating a counterexample point where the Lyapunov conditions fail, and *synthesis* from a finite set of counterexamples generated in the past. We demonstrate that the two steps can be performed using mixed integer linear programming problems (MILP) although no termination guarantees are possible. We show that the branch and cut approach used inside a MILP solver can be adapted to yield a termination guarantee. Although the resulting approach is computationally expensive, it has the advantage of not requiring a "demonstrator" or a pre-existing controller. We provide an empirical evaluation that explores the results of this approach over a set of numerical examples.

ACM Reference Format:
Sara Kamali, Guillaume Berger, and Sriram Sankaranarayanan. 2025. Polyhedral Control Lyapunov Functions for Switched Affine Systems. In *28th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3716863.3718048

1 Introduction

Synthesizing provably correct stabilizing feedback control law is of paramount important in many (safety-critical) applications involving dynamical systems. Many different approaches have been proposed in the literature, including symbolic control [33], set-theoretic methods [8], and co-design of a control feedback law and an associated stability certificate [10, 13]. In this work, we consider the approach of synthesizing a *control Lyapunov function* (CLF). A CLF is a function V satisfying that for every state $\mathbf{x}$, there is an input $\mathbf{u}$ such that V decreases instantly when the system flows from $\mathbf{x}$ with $\mathbf{u}$ applied as input to the system. CLFs have received a lot of attention in the literature, since the seminal work of Sontag [31].

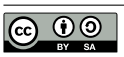
In this paper we explore the problem of synthesizing polyhedral CLFs which are expressed as the maximum over a finite set of linear functions: $V(\mathbf{x}) = \max_{i=1}^{k} \mathbf{c}_i^\top \mathbf{x}$ for a plant model that is specified as a switched affine control system with finitely many modes and a set of affine differential equations corresponding to each mode. Synthesizing polyhedral CLFs is known to be a hard problem. In fact, this is the case even for polyhedral LFs. Kousoulidis et al [21] present an approach for synthesizing polyhedral LFs for linear hybrid systems that requires solving bilinear (nonconvex) constraints.

Our approach is based on the idea of counterexample-guided inductive synthesis (CEGIS), that alternates between finding a candidate CLF and verifying it. Failure to verify a given candidate results in generation of new constraints that restrict the space of future candidate CLFs. This process is iterated until no candidate CLFs remain or a candidate CLF satisfies the verification conditions yielding the required CLF. Counterexample-guided inductive synthesis (CEGIS) provides an avenue to solve the tractability problem posed by the infinite number of constraints while keeping in some cases a form of completeness.

CEGIS approaches for synthesizing LFs have received a lot of attention in the literature, using various templates for the LF (polyhedral, polynomial, Neural Networks, etc.) and approaches for the computation of the candidates and the counterexamples (Linear Program, Mixed-Integer Programming, Sum-of-Squares Optimization, SMT solvers, etc.); see Section 1.1 for a detailed discussion of these approaches. The problem of synthesizing a *CLF* V for a given dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ is in general even more challenging than the one of synthesizing a LF since the condition on V at each $\mathbf{x}$ is a disjunction of several (possibly infinitely many) constraints, namely:

$$(\forall\, \mathbf{x} \neq \mathbf{0})\ (\exists\, \mathbf{u}) \quad L_{f,\mathbf{u}} V(\mathbf{x}) \triangleq \nabla V(\mathbf{x}) \cdot f(\mathbf{x}, \mathbf{u}) < 0.$$

Hence, when designing CEGIS approaches to compute CLFs, we need to account for the fact that even for a small sample set of constraints, the computation of the candidate can be very challenging due to the disjunctive nature of the constraints. One way to address this is to assume a *demonstrator*, which is an oracle that for each counterexample $\mathbf{x}$ provides an input $\mathbf{u}$, thereby "selecting" which clause of the disjunction must be satisfied. The drawback is that if



HSCC '25, Irvine, CA, USA

ACM ISBN 979-8-4007-1504-4/2025/05
https://doi.org/10.1145/3716863.3718048

the demonstrator is not stabilizing, the CEGIS loop might fail to find a CLF. See Section 1.1 for related works on CEGIS approaches for CLF synthesis.

In this paper, we consider *switched affine controlled systems*, that is systems that have a finite set of modes q, where each mode q has a domain and a finite set of dynamics $\dot{\mathbf{x}} = A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}$, $j = 1, \ldots, l$. The control problem is to select for each pair $(q, \mathbf{x})$ a dynamic j such that the resulting *switched affine system* is (strongly) stable for all the solutions. Switched affine controlled systems arise naturally in a wide range of applications [25, 32], or as approximations of nonlinear systems [23]. We also consider polyhedral CLFs, which are convex piecewise linear CLFs, with a pre-defined number of linear pieces. We show that for these type of systems and CLFs, we can formulate the computation of the candidates and the counterexamples as Mixed-Integer Linear Programs (MILPs). This allows us to leverage the power of MILP solvers to solve these problems in an efficient manner. In particular, the MILP solvers we used are guaranteed to find a feasible point if one exists. This allows us to prove an asymptotic completeness guarantees on our CEGIS framework. By contrast, other approaches using Sum-of-Squares relaxations or Neural Networks generally do not guarantee completeness because of the added conservatism or the presence of local minima.

As part of the asymptotic completeness guarantee, we prove an upper bound on the number of iterations of the CEGIS loop. The bound can be very large, and we observed in numerical experiments that the actual number of iterations is much smaller. Therefore, we propose also an alternative approach, that is computationally more efficient, but does not come with a formal bound on the number of iterations. We demonstrate the validity and efficiency of our approach on several benchmark examples.

We examine the performance of our approach on a set of small numerical examples. One limitation of our approach is the need of solving MILPs at each iteration of the CEGIS with a number of integer variables growing linearly with the number of iterations. Although, "warm-start" of the MILP solvers using solutions found at the previous iterations already allows us to improve significantly the practical efficiency of the framework, in future work we plan to improve this efficiency by combining it with educated guesses (e.g., using a demonstrator) provided as soft constraints or warm-starts to the solver.

1.1 Related Work

CEGIS frameworks for synthesizing LFs for closed-loop systems have received a lot of attention in the literature in recent years [1, 2, 6, 7, 10, 12, 19, 34]. Our work differs from those ones in that we consider the problem of controller synthesis (not only verification).

CEGIS frameworks for synthesizing controllers have also received attention in the literature in recent years. The works [9, 13, 16] propose counterexample-based approaches to train simultaneously two neural networks: one representing the control feedback function and one representing a Lyapunov function for the closed-loop systems. In [9, 16], SMT solvers (such as Z3 [15] or dREAL [17]) are used to falsify the candidate functions and generate counterexamples. The falsification method in [13] is more similar to ours in that it uses MILP, thriving on the fact that the functions are PWA, arising from ReLU neural networks. Our work differs from the above in that we do not learn a control feedback function during training; instead we learn a CLF, from which a stabilizing feedback control in derived afterwards. Also the frameworks in [9, 13, 16] do not come in general with formal guarantees of correctness, as they can be stucked in local minima.

The works [14, 29] propose a counterexample-guided framework to learn polynomial CLF for switched systems. SMT solvers and Sum-of-Squares (SoS) programming are used to generate the candidate CLFs, while SoS programming is used to generate the counterexamples. One limitation of this approach is that the SoS relaxation introduces conservatism in the method. By contrast, our work focuses on switched affine systems and polyhedral CLFs in order to provide a sound and asymptotically complete procedure by leveraging the efficiency of MILP solvers to find global solutions to mixed-integer linear problems.

The work [30] proposes a counterexample-guided approach to controller synthesis that assumes a demonstrator, on top of the learner and the falsifier, to provide a control input for every counterexample. This allows to remove the need of SMT solvers in the generation of the candidate CLF. However, if the demonstrator is not stabilizing, then the approach might fail. By contrast, our approach does not require a demonstrator; hence, removing the burden of synthesizing a priori a stabilizing controller.

The work [20] synthesizes likely CLFs for robotic tasks based on demonstrations without relying on a verifier or counterexamples to refine the solution. Their method learns a Lyapunov function from a set of sampled states, models motion estimates using regression techniques, and ensures stability by solving a convex optimization problem. Because their approach lacks formal verification steps, the correctness of the derived CLF is not guaranteed formally, which contrasts with methods that iteratively refine CLFs using counterexamples to ensure formal stability guarantees.

The works [3, 5–8, 18, 21, 24, 26, 28] focus on the computation of polyhedral LFs for linear or PWA systems. Except for [5, 8, 21], they do not study the problem of controller synthesis. The work [8] uses set propagation to find polyhedral control invariant sets for linear systems. A limitation of such set-theoretic approaches is that the complexity of the propagated set grows very quickly, especially when applied to PWA systems ("mode explosion"). The work [5] proposes an approach to synthesize CLFs based on triangulations of the state space. In order to guarantee that a CLF can be found, the triangulation must be fine enough, which induces an exponential complexity of the approach with the dimension of the system. By contrast, our counterexample-based approach keeps a fixed complexity of the CLF throughout all iterations. The work [21] proposes a nonconvex bilinear optimization problem for the computation of the CLF, which is solved using alternating minimization (block-coordinate descent), without guarantees of global optimality (i.e., completeness). The works [6, 7] connect to ours in that they also use counterexample-based approaches. However, they are restricted to the verification of closed-loop systems.

2 Preliminaries

In this section, we define the model of switched systems, introduce the class of piecewise affine functions and a candidate piecewise affine control Lyapunov function (CLF) along with its associated

control feedback law. We recall previously known results from nonsmooth analysis to show that a candidate piecewise affine CLF along with the canonical feedback law yields an asymptotically stable system.

We use capital letters $A, B, C, \ldots$ to denote matrices and boldface font $\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{u}$ to denote vectors. For $m \in \mathbb{N}$, we let $[m] = \{1, \ldots, m\}$. $\bigvee$ and $\bigwedge$ denote *logical or* and *logical and* respectively. $\mathrm{cl}(\cdot)$ shows a closure of a set.

Let $X \subseteq \mathbb{R}^n$ denote the state-space and $\mathbf{x}^* \in X$ be a desired equilibrium point. Throughout this paper, we take $\mathbf{x}^* = \mathbf{0}$ without loss of generality.

Definition 2.1 (Nondeterministic Switched Affine System). A continuous time nondeterministic switched affine system (NSAS) over a state-space X is a tuple $\Gamma = \langle Q, \mathcal{I}, \mathcal{D} \rangle$ wherein Q is a *finite* set of modes, $\mathcal{I}$ maps each mode $q \in Q$ to a domain $\mathcal{I}(q) \subseteq X$ and $\mathcal{D}$ maps each mode $q \in Q$ to a pair $\mathcal{D}(q) = (A_q, \mathbf{b}_q)$ so that the dynamics associated with the mode q are given by $\dot{x} = A_q\mathbf{x} + \mathbf{b}_q$.

The system is *polyhedral* if for each $q \in Q$, $\mathcal{I}(q)$ is defined by a convex polyhedron of the form $C_q\mathbf{x} \leq \mathbf{d}_q$.

We assume that $\bigcup_{q \in Q} \mathcal{I}(q) = X$, i.e, each state belongs to some mode $q \in Q$. For given mode q, we denote the dynamics $\mathcal{D}(q)$ by matrices $(A_q, \mathbf{b}_q)$. However, a state may belong to more than one mode. We can view a nondeterministic switched affine system as a differential inclusion of the form:

$$\dot{\mathbf{x}} \in \mathrm{convex}\,\{A_q\mathbf{x} + \mathbf{b}_q \mid q \in Q,\ \mathbf{x} \in \mathcal{I}(q)\},$$

wherein $\mathrm{convex}(F)$ denotes the set of all convex combinations for a finite set of vectors F. A trajectory of the system is given by a *Carathéodeory* solution to this differential inclusion, which is well defined since we use the convex hull over the dynamics corresponding to all the modes corresponding to a given state [4, Theorem 2.3].

Our plant model is given as a continuous-time switched affine control system wherein each mode is associated with multiple dynamics from which the controller chooses one based on the current state and mode.

Definition 2.2 (Switched Affine Control System). A continuous time switched affine control system (SACS) has a *finite* set of modes Q, wherein each mode q is associated with the following information: (a) a domain $\mathcal{I}(q) \subseteq X$, and (b) a *finite* set of possible dynamics $\mathcal{D}(q) = \{(A_{q,1}, \mathbf{b}_{q,1}), \ldots, (A_{q,l}, \mathbf{b}_{q,l})\}$, wherein $(A_{q,i}, \mathbf{b}_{q,i})$ represents the ODE: $\dot{\mathbf{x}} = A_{q,i}\mathbf{x} + \mathbf{b}_{q,i}$.

We say that the control system is *polyhedral* if each $\mathcal{I}(q)$ is expressed as a convex polyhedron $C_q\mathbf{x} \leq \mathbf{d}_q$.

Note that two modes may overlap: $\mathcal{I}(q_1) \cap \mathcal{I}(q_2) \neq \emptyset$ for some $q_1 \neq q_2$. We now list key simplifying assumptions:

Assumption 1. *We will assume the following structural properties for any SACS we consider in this paper:*

1. *Every state belongs to at least one mode:* $\bigcup_{q \in Q}^{m} \mathcal{I}(q) = X$,
2. *If* $\mathbf{0} \in \mathcal{I}(q)$ *for a mode* $q \in Q$, *then for each available dynamics* $(A_{q,i}, \mathbf{b}_{q,i}) \in \mathcal{D}(q)$, *it holds that* $\mathbf{b}_{q,i} = \mathbf{0}$.[1]
3. *For all* $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$, $\mathbf{0} \notin \{A_{q,i}\mathbf{x} + \mathbf{b}_{q,i} \mid (A_{q,i}, \mathbf{b}_{q,i}) \in \mathcal{D}(q)\}$. *This condition is not strictly necessary. Nevertheless, it simplifies the analysis of the algorithms we will present subsequently in the paper.*
4. $|\mathcal{D}(q)| = l$ *for all* $q \in Q$. *In other words, all modes have the same number of available dynamics. This assumption is not strictly necessary. It serves to make the presentation simpler.*

[1] This condition may be relaxed by requiring one of the dynamics to have zero affine component.

Example 2.3. Consider an example SACS Π with $Q = \{q_1, q_2\}$ and $n = 2$. We have $\mathcal{I}(q_1) = \{\mathbf{x} \mid \mathbf{x}_1 \leq 0\}$ and $\mathcal{I}(q_2) = \{\mathbf{x} \mid \mathbf{x}_1 \geq 0\}$.

Each mode has two dynamics associated with it ($l = 2$). The dynamics associated with q_1 are

$$A_{q_1,1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad A_{q_1,2} = \begin{pmatrix} 0.1 & -0.8 \\ 1.3 & 0 \end{pmatrix}, \quad \mathbf{b}_{q_1,1} = \mathbf{b}_{q_1,2} = \mathbf{0}\,.$$

The dynamics associated with q_2 are

$$A_{q_2,1} = \begin{pmatrix} -0.1 & 0.1 \\ 0.5 & -0.05 \end{pmatrix}, \quad A_{q_2,2} = \begin{pmatrix} -0.1 & 0.1 \\ -0.5 & -1.0 \end{pmatrix}, \quad \mathbf{b}_{q_2,1} = \mathbf{b}_{q_2,2} = \mathbf{0}\,.$$

The eigenvalues of $A_{q_1,1}$ are $\pm i$, $A_{q_1,2}$ are $0.05 \pm 1.02i$, $A_{q_2,1}$ are $-0.3, 0.15$ and $A_{q_2,2}$ are $-0.94, -0.16$. In other words, $A_{q_2,2}$ is a stable dynamic but applicable only when $x_1 \geq 0$.

Definition 2.4 (Switching Feedback Law). A (state-based) switching feedback law is defined by a sequence of maps $\sigma = \langle \sigma(q) \rangle_{q \in Q}$, one for each state $q \in Q$, wherein $\sigma(q) : \mathcal{I}(q) \mapsto [l]$ maps each state $\mathbf{x}$ that belongs to mode $q \in Q$ to the index of one of the dynamics associated with the mode in the set $\mathcal{D}(q)$. For convenience given a mode $q \in Q$ and a state $\mathbf{x} \in \mathcal{I}(q)$, we write $\sigma(q, \mathbf{x})$ as a short-hand for $\sigma(q)(\mathbf{x})$.

We say that the feedback law is *polyhedral* if for each $q \in Q$ and $i \in [l]$, the *closure of the* set $\{\mathbf{x} \mid \sigma(q, \mathbf{x}) = i\}$ can be written as a finite union of convex polyhedra $\bigcup_{k \in \{1,\ldots,N(q,i)\}} G_{q,i,k}\mathbf{x} \leq \mathbf{h}_{q,i,k}$ for some natural number $N(q, i)$.

Definition 2.5 (Composition of SACS and Feedback Law). Given a SACS Π and a switched feedback law σ, we can define their composition as a NSAS whose discrete modes are given by the set $\hat{Q} = \{(q, i) \mid q \in Q, i \in [l]\}$ wherein the mode (q, i) is associated with the dynamics $\dot{\mathbf{x}} = A_{q,i}\mathbf{x} + \mathbf{b}_{q,i}$ and domain $\mathcal{I}(q, i)$ is the set $\mathrm{cl}\,(\{\mathbf{x} \mid \sigma(q, \mathbf{x}) = i\})$.

If a feedback law σ is polyhedral, then we can express the composition as a polyhedral NSAS whose modes are written as $\hat{Q} = \{(q, i, k) \mid q \in Q, i \in [l], k \in [N(q, i)]\}$ such that $\mathcal{I}(q, i, k)$ is the convex polyhedron $G_{q,i,k}\mathbf{x} \leq \mathbf{h}_{q,i,k}$.

2.1 Polyhedral Control Lyapunov Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *polyhedral* if it is of the form $f(\mathbf{x}) = \max_{i \in [k]} \mathbf{c}_i^\top \mathbf{x}$. Such a function is said to have k pieces. The function is *positive definite* if for all $\mathbf{x} \in X \setminus \{\mathbf{0}\}$, $f(\mathbf{x}) > 0$. Throughout the paper, we will assume $k \geq 2$.

Definition 2.6 (Lyapunov Function). Let $\Gamma : \langle Q, \mathcal{I}, \mathcal{D} \rangle$ be a given non-deterministic switched system. A polyhedral function $V(\mathbf{x}) = \max_{i \in [k]} \mathbf{c}_i^\top \mathbf{x}$ is said to be a *Lyapunov function* for Γ if (a) $V(\mathbf{x})$ is positive definite, and (b) for all $q \in Q$ and $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$, the following condition holds:

$$(\forall\, i \in [k]) \quad V(\mathbf{x}) = \mathbf{c}_i^\top \mathbf{x} \implies \mathbf{c}_i^\top (A_q\mathbf{x} + \mathbf{b}_q) < 0. \tag{1}$$

Theorem 2.7 ([8, Theorem 2.19]). *If* $V(\mathbf{x})$ *is a polyhedral Lyapunov function for a switched system* Γ, *then* Γ *is globally asymptotically stable.*

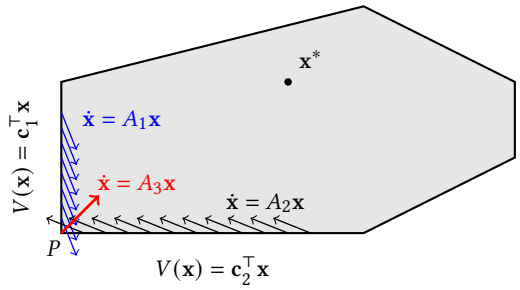


Figure 1: Example that shows designing the feedback switching law based on Eq. (3) may fail to achieve stability.

We now extend our approach from Lyapunov functions for nondeterministic switched systems to SACS which allow one of l possible dynamics for each mode q. Let $\Pi : \langle Q, \mathcal{I}, \mathcal{D}\rangle$ be a SACS with $|Q| = m$ modes and $l \geq 2$ dynamics for each mode that satisfies Assumption 1.

Definition 2.8 (Control Lyapunov Function). A polyhedral function $V(\mathbf{x}) = \max_{i\in[k]} \mathbf{c}_i^\top\mathbf{x}$ is a *control Lyapunov function* (CLF) for a SACS Π if (a) $V(\mathbf{x})$ is positive definite, and (b) for each $q \in Q$ and $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$:

$$(\exists\, j \in [l])\ (\forall\, i \in [k]) \quad V(\mathbf{x}) = \mathbf{c}_i^\top\mathbf{x} \implies \mathbf{c}_i^\top(A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) < 0. \tag{2}$$

In other words, for every state $\mathbf{x}$ and mode q, there is a choice of dynamics from the set of available ones that ensures that all the pieces of the CPWL that are maximized at the state $\mathbf{x}$ can be decreased by the chosen dynamics.

Now, the question is how can we define the switching signal as a feedback law. We can make the choice arbitrarily by simply picking one of the many possible choices that result in a decrease:

$$\sigma(q, \mathbf{x}) = \min\{j \in [l] \mid j \text{ satisfies condition (2) for } (q, \mathbf{x})\}. \tag{3}$$

However, such an approach does not always result in a feedback law that yields a stable trajectory. The key here is that we are forced to define dynamics on regions of measure 0 whose neighborhoods have different dynamics. Using the Carathéodeory solution allows us to ignore these regions effectively, possibly leading to unstable behaviors. Consider the situation shown in Figure 1: the dynamics A_3 is chosen by the switching rule only at point P and the two other dynamics that are chosen in the neighborhood of P do not satisfy the decrease condition at point P. If we choose the active subsystem based on Eq. (3), the decreasing condition does not hold. To address this problem, we will define a canonical feedback law in Section 2.2. This feedback is chosen carefully to yield a feedback law $\sigma(q, \mathbf{x})$ such that for each $j \in [l]$, the decrease condition holds for the closure of the set $\{\mathbf{x} \mid \sigma(q, \mathbf{x}) = j\}$. This will avoid the situation shown in Figure 1.

2.2 Switching Feedback Law from CLF

Let $V(x) = \max_{i\in[k]} \mathbf{c}_i^\top x$ be a CLF for a SACS Π. We will now derive a feedback law $\sigma(q, x)$ for mode q and state $x \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$. We wish the feedback law to have the following *closure property*.

Definition 2.9 (Closure Property). For any $q \in Q$ and $j \in [l]$, let $\mathcal{D}(q, j) = \{\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\} : \sigma(q, \mathbf{x}) = j\}$. We say that the feedback σ has the *closure property* iff for all $\mathbf{x} \in \mathrm{cl}(\mathcal{D}(q, j)) \setminus \{\mathbf{0}\}$,

$$(\forall\, i \in [k]) \quad V(\mathbf{x}) = \mathbf{c}_i^\top\mathbf{x} \implies \mathbf{c}_i^\top(A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) < 0. \tag{4}$$

Seen another way, if a feedback law has a closure property then for any convergent sequence of states $\mathbf{x}_1, \cdots, \mathbf{x}_j, \cdots \to \mathbf{x}^*$ all belonging to the set $\mathcal{I}(q) \setminus \{\mathbf{0}\}$, if $\sigma(q, \mathbf{x}_i) = j$ for each $\mathbf{x}_i$, then $\mathbf{x}^*$ satisfies the decrease condition (4) for dynamics j even if $\sigma(q, \mathbf{x}^*) \neq j$. A feedback law with closure property can therefore avoid the situation shown in Figure 1, since the vertex P belongs to the closure the region wherein the feedback maps states to the dynamics A_1 (or A_2). Therefore, A_1 (or A_2) must necessarily satisfy the decrease condition at P, if the feedback law satisfies the closure property.

Our strategy is to define the choice of dynamics at each state by means of a *merit function* $\epsilon(q, \mathbf{x}, j)$ for $q \in Q$, $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$ and $j \in [l]$, satisfying the following properties:

(1) $\epsilon(q, \mathbf{x}, j) \geq 0$ for all $q, \mathbf{x}, j$ (belonging to the respective sets, as stated above).
(2) $\epsilon(q, \mathbf{x}, j) > 0$ if and only if the dynamics $\dot{\mathbf{x}} = A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}$ satisfies the decrease condition (4) for mode q and state $\mathbf{x}$.
(3) For fixed q and j, $\epsilon(q, \mathbf{x}, j)$ is a continuous function in $\mathbf{x}$, for all $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$.

If a function $\epsilon(q, \mathbf{x}, j)$ can be defined for each mode $q \in Q$, $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$ and $j \in [l]$, we can derive a feedback law

$$\sigma(q, \mathbf{x}) := \mathrm{argmax}_{j\in[l]}\, \epsilon(q, \mathbf{x}, j). \tag{5}$$

Theorem 2.10. *The feedback law $\sigma(q, \mathbf{x})$ derived from a merit function $\epsilon(q, \mathbf{x}, j)$ following (5) satisfies the closure property, i.e, for all $\mathbf{x} \in \mathrm{cl}(\mathcal{D}(q, j)) \setminus \{\mathbf{0}\}$, the decrease condition (4) holds.*

Proof. Let $\mathbf{x} \in \mathrm{cl}(\mathcal{D}(q, j)) \setminus \{\mathbf{0}\}$. Due to the fact that $V(\mathbf{x})$ is a CLF, there exists a mode $p \in [l]$ (p may or may not be the same as j) such that the decrease condition holds at $\mathbf{x}$ for p, and thus, $\epsilon(q, \mathbf{x}, p) > 0$ by property (2). Since $\epsilon(q, \mathbf{x}, p)$ is continuous over $\mathbf{x}$ by property (3), we deduce the existence of an open neighborhood U of $\mathbf{x}$ and a constant $c > 0$ such that $\epsilon(q, \mathbf{y}, p) > c$ for all $\mathbf{y} \in U$. Hence, for $\mathbf{y} \in \mathcal{D}(q, j) \cap U$, we have $\epsilon(q, \mathbf{y}, j) \geq \epsilon(q, \mathbf{y}, p) > c$. Therefore, applying property (3), we conclude that $\epsilon(q, \mathbf{x}, j) \geq c > 0$. Therefore, the decrease condition (4) for $j \in [l]$ holds for all $\mathbf{x} \in \mathrm{cl}(\mathcal{D}(q, j)) \setminus \{\mathbf{0}\}$. □

It remains to define a merit function that satisfies the properties noted above. There are many ways to define such a function, we consider one such definition. For convenience, let us denote $v = V(\mathbf{x})$, $v_i = \mathbf{c}_i^\top\mathbf{x}$ and $\hat{v}_i = \mathbf{c}_i^\top(A_{q,j}\mathbf{x} + \mathbf{b}_{q,j})$. We define ϵ as

$$\epsilon(q, \mathbf{x}, j) = \min_{i\in[k]} \max\{\tau \geq 0 \mid (v - v_i) - \tau\hat{v}_i - \tau^2 \geq 0\}.$$

Clearly $\epsilon(q, \mathbf{x}, j) \geq 0$ for all $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$.

Theorem 2.11. *$\epsilon(q, \mathbf{x}, j) > 0$ if and only if $(\forall\, i \in [k])\ V(\mathbf{x}) = \mathbf{c}_i^\top\mathbf{x} \implies \mathbf{c}_i^\top(A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) < 0$.*

Proof. ($\Longleftarrow$) Let us assume that the decrease condition holds: $(\forall\, i \in [k])\ v = v_i \implies \hat{v}_i < 0$. We wish to show that $\epsilon(q, \mathbf{x}, j) > 0$. Consider two cases: (a) $v = v_i$ or (b) $v_i < v$. Note that $v = \max(v_1, \ldots, v_k)$ and therefore, no other case is possible.

For case (a), we note that $\max\{\tau \geq 0 \mid (v-v_i)-\tau\hat{v}_i-\tau^2 \geq 0\} > 0$ since $v = v_i$, we have $\hat{v}_i < 0$. Therefore, $-\tau\hat{v}_i - \tau^2 \geq 0$ if and only if $\tau \in [0, -\hat{v}_i]$ and by the decrease condition, $\hat{v}_i < 0$.

For case (b), the polynomial $(v - v_i) - \tau\hat{v}_i - \tau^2 > 0$ when $\tau = 0$ since $v > v_i$. It remains non-negative for $\tau \in [0, r_i]$ wherein $r_i = \frac{-\hat{v}_i+\sqrt{\hat{v}_i^2+4(v-v_i)}}{2}$ is a positive root of the polynomial. Therefore, $\max\{\tau \geq 0 \mid (v - v_i) - \tau\hat{v}_i - \tau^2 \geq 0\} = r_i > 0$. We conclude that $\epsilon(q, \mathbf{x}, j) > 0$ since it is the minimum over k positive terms.

($\Longrightarrow$) Conversely, let us assume that the decrease condition fails to hold at $\mathbf{x}$ for mode j. Therefore, $\exists i \in [k]$ such that $v = v_i$ and $\hat{v}_i \geq 0$. The polynomial $(v - v_i) - \tau\hat{v}_i - \tau^2 = -\tau\hat{v}_i - \tau^2 < 0$ for all $\tau > 0$. Therefore, $\max\{\tau \geq 0 \mid (v - v_i) - \tau\hat{v}_i - \tau^2 \geq 0\} = 0$. Hence, $\epsilon(q, \mathbf{x}, j) = 0$. □

Theorem 2.12. *$\epsilon(q, \mathbf{x}, j)$ is continuous over $\mathbf{x} \in \mathcal{I}(q) \setminus \{\mathbf{0}\}$.*

Proof. Note that $\epsilon(q, \mathbf{x}, j)$ can be written as $\min_{i\in[k]} \epsilon_i(q, \mathbf{x}, j)$, wherein $\epsilon_i(q, \mathbf{x}, j) := \max\{\tau \geq 0 \mid (v - v_i) - \tau\hat{v}_i - \tau^2 \geq 0\}$. Clearly, if $v = v_i$, then $\epsilon_i(q, \mathbf{x}, j) = \max(-\hat{v}_i, 0)$. Otherwise, if $v > v_i$, then $\epsilon_i(q, \mathbf{x}, j) = \frac{-\hat{v}_i+\sqrt{\hat{v}_i^2+4(v-v_i)}}{2}$. Thus, each $\epsilon_i(q, \mathbf{x}, j)$ is continuous and $\epsilon = \min_{i\in[k]} \epsilon_i$ is continuous. □

Observe that σ is not necessarily polyhedral since the closure of $\mathcal{D}(q, j)$ is not necessarily the union of finitely many polyhedra. It is possible to modify the function ϵ slightly so that σ is polyhedral if the state-space is compact. We will provide details in an extended version.

3 Verification of Polyhedral CLFs

In this section, we provide algorithms that verify a given polyhedral function $V(\mathbf{x}) = \max_{i\in[k]} \mathbf{c}_i^\top \mathbf{x}$ satisfies the conditions for a CLF for a SACS $\Pi : \langle Q, \mathcal{I}, \mathcal{D}\rangle$ given in Definition 2.8. We show that the conditions can be expressed by checking if a sequence of (mixed-integer) linear programs (MILPs) is infeasible.

Let $C : \langle \mathbf{c}_1, \ldots, \mathbf{c}_k\rangle$ represent the coefficients of a polyhedral function $V(\mathbf{x}) = \max_{i\in[k]} \mathbf{c}_i^\top \mathbf{x}$. We wish to check if it is a CLF for a given SACS Π or find a witness state $\mathbf{x} \in X$ for which the CLF conditions fail.

Verifying positive definiteness: First, we need to verify that $V(\mathbf{x})$ is positive definite. In other words, for all $\mathbf{x} \neq \mathbf{0}$, we wish to check that $V(\mathbf{x}) > 0$. We formulate a feasibility problem that focuses on finding a witness where the condition fails, i.e, $V(\mathbf{x}) \leq 0$. If the problem is infeasible, we note that V is positive definite.

$$\left.\begin{array}{ll} \text{find} & \mathbf{x} \in X \\ \text{s.t.} & \mathbf{c}_i^\top \mathbf{x} \leq 0 \quad i = 1, \ldots, k \\ & \mathbf{x} \neq \mathbf{0} \end{array}\right\} \tag{6}$$

The difficulty lies in enforcing the constraint $\mathbf{x} \neq \mathbf{0}$. This is achieved using a simple trick that notes that $V(\mathbf{x}) \leq 0$ implies that $V(\lambda\mathbf{x}) \leq 0$ for all $\lambda \geq 0$. Let $\delta > 0$ be a positive constant such that the hypercube $H_\delta = [-\delta, \delta]^n$ lies entirely in X.

Lemma 3.1. *Let $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ satisfy $V(\mathbf{x}) \leq 0$. There exists $\mathbf{y} \in \mathrm{bd}(H_\delta)$ such that $V(\mathbf{y}) \leq 0$.*

Proof. Let $\lambda = \delta/\|\mathbf{x}\|_\infty$. It holds that $\lambda\mathbf{x} \in \mathrm{bd}(H_\delta)$. Furthermore, $V(\lambda\mathbf{x}) = \lambda V(\mathbf{x}) \leq 0$. This concludes the proof. □

Thus, verifying if $V(\mathbf{x})$ is positive definite can be solved by formulating $2n$ linear programs, each of which fixing $x_\alpha = \delta$ or $x_\alpha = -\delta$ for some $\alpha \in [n]$.

Verifying the decrease condition: Next, we present an approach to verifying the decrease condition in (2). Rather than verifying this condition, we seek to find a witness $\mathbf{x}$ for its negation and the infeasibility of doing so signals that the condition holds.

Therefore, we seek for each mode $q \in Q$ to solve the following feasibility problem:

$$\left.\begin{array}{ll} \text{find} & \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \mathcal{I}(q), \;\; \mathbf{x} \neq \mathbf{0}, \\ & (\forall\, j \in [l])\; (\exists\, i \in [k]) \\ & \quad \bigwedge_{i'\in[k]} \mathbf{c}_i^\top \mathbf{x} \geq \mathbf{c}_{i'}^\top \mathbf{x} \;\wedge\; \mathbf{c}_i^\top (A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) \geq 0. \end{array}\right\} \tag{7}$$

The negation expresses the idea that for every state $\mathbf{x} \in \mathcal{I}(q)\setminus\{\mathbf{0}\}$ and for each $j \in [l]$ there is a piece $\mathbf{c}_i^\top \mathbf{x}$ that is maximal at $\mathbf{x}$ and fails to satisfy the decrease condition. This can be encoded using $k \times l$ binary variables $w_{j,i}$ for each $i \in [k]$ and $j \in [l]$. The idea is that for each j, $w_{j,i} = 1$ if $i \in [k]$ "witnesses" the lack of decrease using the dynamic $j \in [l]$. Thus, we obtain the equivalent problem:

$$\left.\begin{array}{ll} \text{find} & \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \mathcal{I}(q), \;\; \mathbf{x} \neq \mathbf{0}, \\ & (\forall\, j \in [l]) \;\; \sum_{i=1}^k w_{j,i} = 1, \\ & (\forall\, j \in [l])\; (\forall\, i \in [k]) \;\; w_{j,i} = 1 \implies \\ & \quad \left\{\bigwedge_{i'\in[k]} \mathbf{c}_i^\top \mathbf{x} \geq \mathbf{c}_{i'}^\top \mathbf{x} \;\wedge\; \mathbf{c}_i^\top (A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) \geq 0\right\}. \end{array}\right\} \tag{8}$$

The last constraint is a *conditional constraint* which can be encoded as a MILP constraint using the "big-M" trick [35].

Suppose $\mathbf{0} \notin \mathcal{I}(q)$, the constraint $\mathbf{x} \neq \mathbf{0}$ can be ignored. Otherwise, we can run $2n$ separate MILPs by fixing $x_\alpha = \pm\delta$ for each $\alpha \in [n]$. The $2n$ MILPs are succinctly represented as:

$$\left.\begin{array}{ll} \text{find} & \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \mathcal{I}(q), \;\; \mathbf{x} \neq \mathbf{0}, \\ & (\forall\, j \in [l]) \;\; \sum_{i=1}^k w_{j,i} = 1, \\ & (\forall\, j \in [l])\; (\forall\, i \in [k]) \;\; w_{j,i} = 1 \implies \\ & \quad \left\{\bigwedge_{i'\in[k]} \mathbf{c}_i^\top \mathbf{x} \geq \mathbf{c}_{i'}^\top \mathbf{x} \;\wedge\; \mathbf{c}_i^\top (A_{q,j}\mathbf{x} + \mathbf{b}_{q,j}) \geq 0\right\}, \\ & \bigvee_{\alpha\in[n]} x_\alpha = \delta \;\vee\; \bigvee_{\alpha\in[n]} x_\alpha = -\delta. \end{array}\right\} \tag{9}$$

Each disjunct is used in a separate MILP instance.

This works because if $\mathbf{0} \in \mathcal{I}(q)$, we have already assumed that $\mathbf{b}_q = \mathbf{0}$ (Assumption 1). This means that if a particular solution $\mathbf{x} \neq \mathbf{0}$ satisfies (8) then so does $\lambda\mathbf{x}$ for $\lambda > 0$ and $\lambda\mathbf{x} \in \mathcal{I}(q)$.

Therefore, verification of condition (2) is achieved by means of running at most $2n$ MILP instances with $k \times l$ binary variables, n continuous variables and $O(kl)$ constraints. If feasible, it yields a witness $\mathbf{x}$ for which the decrease condition fails. Otherwise, all the MILPs so obtained are infeasible, we conclude that the given CLF V is valid.

4 Synthesis of Polyhedral CLFs

In this section, we describe a counterexample-guided approach to synthesizing polyhedral CLFs $V(\mathbf{x}) = \max_{i\in[k]} \mathbf{c}_i^\top \mathbf{x}$ wherein the number of pieces k is fixed *a priori*. We first describe the overall

counterexample driven approach and propose two algorithms based on this approach.

We assume as inputs a SACS $\Pi : \langle Q, \mathcal{I}, \mathcal{D} \rangle$, a bound $k \geq 2$ on the number of pieces of the polyhedral CLF that we seek and a bound $K > 0$ on each coefficient. We assume a user-input $\alpha > 0$ which we call the *robustness parameter*: we will demonstrate how it can be used to prove convergence of our iterative scheme. The approach searches for unknown coefficients $C : \langle \mathbf{c}_1, \ldots, \mathbf{c}_k \rangle$, wherein $\mathbf{c}_i \in \mathbb{R}^n$. We also impose the constraint $-K\mathbf{1} \leq \mathbf{c}_i \leq K\mathbf{1}$, i.e, each entry of $\mathbf{c}_i$ lies in the interval $[-K, K]$, where K is given as an input.

The counterexample-driven synthesis approach is based on the alternation between two phases: learning and verification. At the r^{th} iteration, we maintain a constraint $\Psi_r[\mathbf{c}_1, \ldots, \mathbf{c}_k]$ that represents the unexplored solutions for the coefficients of the desired CLF.

- To begin with, $\Psi_0 : \bigwedge_{i\in[k]} -K\mathbf{1} \leq \mathbf{c}_i \leq K\mathbf{1}$.
- At each iteration, we check if Ψ_r is feasible. If so, we obtain a feasible *candidate* solution $C_r : \langle \mathbf{c}_{r,1}, \ldots, \mathbf{c}_{r,k} \rangle$ that we use to instantiate the candidate polyhedral function $V_r(\mathbf{x}) = \max_{i\in[k]} \mathbf{c}_{r,i}^\top \mathbf{x}$.
- We verify using the techniques from Section 3 whether $V_r(\mathbf{x})$ is a valid polyhedral CLF.
- If the verification succeeds, we have our CLF $V(\mathbf{x}) = V_r(\mathbf{x})$ and we terminate successfully.
- Otherwise, we have a witness $\mathbf{x} = \mathbf{x}_r$ where the approach fails. We will use this witness to eliminate C_r and possibly other candidates from further consideration.

We propose two approaches for incorporating witnesses: (a) a *direct approach* wherein the witness $\mathbf{x}_r$ to create a formula $\Psi_{r+1} = \Psi_r \wedge \Psi_{\text{block}}(\mathbf{x}_r)$; (b) a *tree-based search* wherein, we maintain Ψ_r as a tree whose nodes are labeled with polyhedra over the unknown coefficients C. In this approach, every branch of the tree may generate a different set of witnesses. We will show using the robustness parameter $\alpha > 0$ that the tree approach terminates although the computational complexity can be quite high.

4.1 Direct Counterexample-Driven Approach

We add a constraint based on the witness: $\Psi_{r+1} = \Psi_r \wedge \Psi_{\text{block}}(\mathbf{x}_r)$, wherein the so-called *blocking assertion* $\Psi_{\text{block}}(\mathbf{x}_r)$ depends on the witness $\mathbf{x}_r$ and is described below. The goal of $\Psi_{\text{block}}(\mathbf{x}_r)$ is to ensure that the candidate C_r, which failed at $\mathbf{x}_r$, is removed from the space of unexplored solutions. The blocking assertion is composed of two conjunctive assertions, i.e., $\Psi_{\text{block}}(\mathbf{x}_r) = \Psi_1(\mathbf{x}_r) \wedge \Psi_2(\mathbf{x}_r)$, where Ψ_1 and Ψ_2 are described below. We drop the $\mathbf{x}_r$ from the notation for the sake of readability. First, Ψ_1 postulates that some piece $i \in [k]$ is maximized at $\mathbf{x}_r$ and the function V_r evaluated at $\mathbf{x}_r$ is greater than equal to $\alpha\|\mathbf{x}_r\|$, wherein α is a user-provided "robustness" parameter:

$$\Psi_1[C] : \bigvee_{i\in[k]} \bigwedge_{i'\in[k]} \mathbf{c}_i^\top \mathbf{x}_r \geq \mathbf{c}_{i'}^\top \mathbf{x}_r \ \wedge\ \mathbf{c}_i^\top \mathbf{x}_r \geq \alpha\|\mathbf{x}_r\| \,.$$

The term $\alpha\|\mathbf{x}_r\|$ is used in place of a strict positivity constraint $\mathbf{c}_i^\top \mathbf{x}_r > 0$ and without loss of generality, we can use the L_2-norm $\|\mathbf{x}_r\|^2 = \mathbf{x}_r^\top \mathbf{x}_r$. We can encode this as a combination of real-valued and binary indicator variables $w_{r,1}, \ldots, w_{r,k}$, wherein $w_{r,i}$ is used to indicate the that i^{th} piece of the polyhedral function is larger than the others at $\mathbf{x}_r$:

$$\Psi_1[C, \mathbf{w}_r] : \begin{cases} \sum_{i\in[k]} w_{r,i} \geq 1 \\ \bigwedge_{i\in[k]}\ w_{r,i} = 1 \implies \{\bigwedge_{i'\in[k]} \mathbf{c}_i^\top \mathbf{x}_r \geq \mathbf{c}_{i'}^\top \mathbf{x}_r\} \\ \bigwedge_{i\in[k]}\ w_{r,i} = 1 \implies \{\mathbf{c}_i^\top \mathbf{x}_r \geq \alpha\|\mathbf{x}_r\|\} \end{cases}$$

Next, consider each mode $q \in Q$ such that $\mathbf{x}_r \in \mathcal{I}(q)$. Let $\mathcal{D}(q) = \{(A_{q,1}, \mathbf{b}_{q,1}), \ldots, (A_{q,l}, \mathbf{b}_{q,l})\}$. We encode that at least one dynamic causes a strict decrease of a maximal piece at $\mathbf{x}_r$:

$$\Psi_2[C, \mathbf{w}_r] : \bigvee_{j\in[l]} \bigwedge_{i\in[k]} w_{r,i} = 1 \implies \mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \leq -\alpha\|\mathbf{x}_r\| \,.$$

We add new binary variables $s_{r,1}, \ldots, s_{r,l}$, wherein $s_{r,j}$ indicates the dynamics $(A_{q,j}, \mathbf{b}_{q,j})$ are being used. This gives

$$\Psi_2[C, \mathbf{w}_r, \mathbf{s}_r] : \begin{cases} \sum_{j\in[l]} s_{r,j} = 1 \\ \bigwedge_{j\in[l]} \bigwedge_{i\in[k]} (s_{r,j} = 1 \wedge w_{r,i} = 1) \implies \\ \qquad\qquad \mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \leq -\alpha\|\mathbf{x}_r\| \end{cases}$$

We can use the "big-M" trick to write this equivalently as

$$\begin{cases} \sum_{j\in[l]} s_{r,j} = 1 \\ \bigwedge_{j\in[l]} \bigwedge_{i\in[k]} \\ \qquad \mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \leq -\alpha\|\mathbf{x}_r\| + M(2 - s_{r,j} - w_{r,i}) \,. \end{cases}$$

The value of M is estimated as $\alpha\|\mathbf{x}_r\| + K \max_{j\in[l]} \|A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}\|_1$. The blocking constraint is $\Psi_{\text{block}} : \Psi_1[C, \mathbf{w}_r] \ \wedge\ \Psi_2[C, \mathbf{w}_r, \mathbf{s}_r]$.

Lemma 4.1. *The candidate coefficients at the r^{th} iteration $C_r : \langle \mathbf{c}_{r,1}, \ldots, \mathbf{c}_{r,k} \rangle$ do not satisfy $\Psi_{\text{block}}(\mathbf{x}_r)$.*

Proof. Note that the witness $\mathbf{x}_r$ must either demonstrate $V(\mathbf{x}_r) \leq 0$ (violation of positive semidefiniteness) or V fails to satisfy (2) at $\mathbf{x} = \mathbf{x}_r$. $\Psi_1[C]$ enforces that any satisfying solution C must yield a function $V_C(\mathbf{x}_r) \geq \alpha\|\mathbf{x}_r\|$. Clearly $C = C_r$ would violate Ψ_1 and hence Ψ_{block} in this case.

Otherwise, $\mathbf{x}_r$ violates the derivative condition (2). However $\Psi_2[C]$ posits that any solution C must satisfy the derivative condition and $\mathbf{x} = \mathbf{x}_r$. In this case, C_r violates Ψ_2 and hence Ψ_{block}. □

The direct approach starts with an initial formula $\Psi_0[C]$ and at each iteration, it constructs $\Psi_{r+1} = \Psi_r \wedge \Psi_{\text{block}}(\mathbf{x}_r)$ corresponding to the witness $\mathbf{x}_r$ obtained at the r^{th} iteration. This approach can be implemented simply as an alternation between finding candidates by checking Ψ_r for satisfiability and verifying the candidate, each of which can be expressed as a series of MILPs. After r iterations, the MILP for generating candidates will have $O(r(k + ml))$ binary variables (where $|Q| = m$), kn real-valued variables and $O(rk^2 + rml)$ constraints. The time for finding feasibility is bounded by $O(2^{r(k+ml)}\text{poly}(kn, rk^2 + rml))$, wherein $\text{poly}(d, e)$ represents an upper bound on the time taken to find a feasible solution for a linear program with d variables and e constraints.

However, there is no upper bound on the number of iterations r since the process does not have a known convergence guarantee.

Example 4.2. We ran the direct counterexample-driven search procedure on the SACS Π from Example 2.3 with $k = 6$, while setting the bounds $K = 10.0$. The approach yields the CLF below after 97 iterations:

$$V(x_1, x_2) = \max \begin{cases} 10x_1 - 1.53x_2,\ 8.29x_1 + 1.57x_2, \\ -8.29x_1 - 1.57x_2,\ 10x_1 + 1.86x_2, \\ -9.23x_1 + 10x_2,\ -9.66x_1 + 1.61x_2. \end{cases}$$

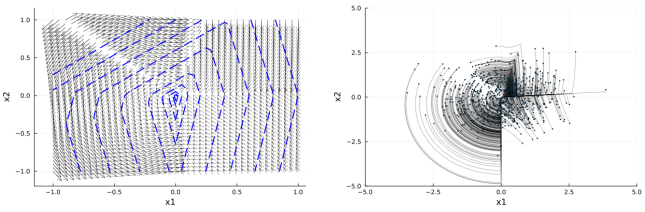


Figure 2: Vector field and trajectories of the closed-loop system shown for the CLF $V(x_1, x_2)$ from Example 4.2.

The computation time was 5 seconds on a Macbook laptop with Apple M3 processor and 24GB RAM. Figure 2 shows the vector field and closed-loop trajectories for a feedback law σ constructed from this CLF.

4.2 Tree-Based Search Algorithm

We now describe a different approach that maintains Ψ_r implicitly as a tree whose nodes are labeled with a set of linear inequalities over C. At the start the tree consists of a single root node N_0 which is labeled with a polyhedron P_0 whose constraints are given by $\Psi_0 : \bigwedge_{i\in[k]} -K\mathbf{1} \leq \mathbf{c}_i \leq K\mathbf{1}$. In practice, we also add a "symmetry breaking" constraint wherein we choose a seed state $\mathbf{x}_\circ \neq \mathbf{0}$ and enforce that $\mathbf{c}_1^\top \mathbf{x}_\circ \geq \alpha\|\mathbf{x}_\circ\|$. At iteration r, we proceed as follows:

(1) Choose a previously unexplored *leaf* node N_r and mark it as explored.
   - If no such node remains in the tree, we exit and declare that no polyhedral CLF could be found.
(2) Select a feasible point C_r that satisfies P_r.
   - If no such point can be found, then we go back to the previous step.
(3) Verify C_r, i.e., try to find a witness.
   - If verification succeeds (i.e., no witness could be found), we terminate having successfully found a CLF C_r.
(4) Otherwise, we have a witness $\mathbf{x}_r$. We use this witness to *refine* the node N_r and add child nodes to N_r in the tree, each associated with a new polyhedron. The refinement process is discussed in detail below.

We define the *refinement* process for a node N_r with polyhedron P_r from a witness $\mathbf{x}_r$. Let $\hat{Q} = \{q \in Q \mid \mathbf{x}_r \in \mathcal{I}(q)\}$ denote the set of *active* modes at $\mathbf{x}_r$. We consider all maps of the form $\mu : \hat{Q} \to [l]$ that assigns the index of a dynamic to each active mode. There are l^{m_r} such maps (where $|\hat{Q}| = m_r$).

We create a child node of the form $\mathsf{N}_{r,\mu,i}$ for each map $\mu : \hat{Q} \to [l]$ and each $i \in [k]$. The rationale is that in node $\mathsf{N}_{r,\mu,i}$, we assign to $(q, \mathbf{x}_r)$ the dynamics $(A_{q,\mu(q)}, \mathbf{b}_{q,\mu(q)})$ and that the i^{th} piece of V is maximal at $\mathbf{x}_r$. Therefore, we have $k \times l^{m_r}$ children. The polyhedron $\mathsf{P}_{r,\mu,i}$ is given as a conjunction $\mathsf{P}_r \wedge \hat{P}_{r,\mu,i}$ wherein $\hat{P}_{r,\mu,i}$ is given by

$$\begin{cases} \bigwedge_{i'\in[k]\setminus\{i\}} \mathbf{c}_i^\top \mathbf{x}_r \geq \mathbf{c}_{i'}^\top \mathbf{x}_r + \alpha\|\mathbf{x}_r\|, \\ \mathbf{c}_i^\top \mathbf{x} \geq \alpha\|\mathbf{x}_r\|, \\ \bigwedge_{q\in\hat{Q}} \mathbf{c}_i^\top (A_{q,\mu(q)}\mathbf{x}_r + \mathbf{b}_{q,\mu(q)}) \leq -\alpha\|\mathbf{x}_r\|. \end{cases} \tag{10}$$

In other words, $\hat{P}_{r,\mu,i}$ forces the i^{th} piece $\mathbf{c}_i^\top \mathbf{x}$ to be "robustly" greater than the other pieces, be robustly positive and for each mode q, the dynamic $\mu(q) \in [l]$ is used to enforce the decrease condition also in a robust manner. Here, by "robustness", we mean that strict inequalities of the form $f(\mathbf{x}) > 0$ are replaced by $f(\mathbf{x}) \geq \alpha\|\mathbf{x}_r\|$.

Lemma 4.3. *The candidate $C_r : \langle \mathbf{c}_{r,1}, \ldots, \mathbf{c}_{r,k} \rangle$ does not belong to the polyhedron $\hat{P}_{r,\mu,i}$ for all $\alpha > 0$.*

Proof. Note that C_r fails to be verified as a CLF. Let V_r be the corresponding CLF. Therefore (case-a) it fails to be positive definite at $\mathbf{x}_r$, or (case-b) it fails the decrease condition at $\mathbf{x}_r$. In this proof, we simply write $\hat{P} = \hat{P}_{r,\mu,i}$.

Consider (case-a): It holds that $V_r(\mathbf{x}_r) \leq 0$. However, $\hat{P}$ ensures that for any solution C that belongs to it, $V_C(\mathbf{x}_r) \geq \alpha\|\mathbf{x}_r\|$. Hence, C_r does not belong to $\hat{P}$. Therefore, the result follows for this case.

Consider (case-b): Since V_r fails the decrease condition at $\mathbf{x} = \mathbf{x}_r$, there is a mode q such that $\mathbf{x}_r \in \mathcal{I}(q)$ and for all $j \in [l]$, there is a piece $i \in [k]$ such that $\mathbf{c}_i^\top \mathbf{x}_r = V_r(\mathbf{x}_r)$ and $\mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \geq 0$.

Since $\mathbf{x}_r \in \mathcal{I}(q)$, we have that $q \in \hat{Q}$. Let $\mu(q) = j$. Let $i' \in [k]$ be the piece that fails the decrease condition corresponding to j, i.e., $\mathbf{c}_{i'}^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \geq 0$. If $i = i'$, then it follows directly that C_r does not belong to $\hat{P}$. This is because $\hat{P}$ includes the inequality : $\mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r + \mathbf{b}_{q,j}) \leq -\alpha\|\mathbf{x}_r\|$ which directly contradicts.

Otherwise, if $i \neq i'$, then for any solution C that satisfies $\hat{P}$, it holds that $\mathbf{c}_i^\top \mathbf{x}_r \geq \mathbf{c}_{i'}^\top \mathbf{x}_r + \alpha\|\mathbf{x}_r\|$ However, for $C = C_r$, we note that $\mathbf{c}_{r,i'}^\top \mathbf{x}_r \geq \mathbf{c}_{r,i}^\top \mathbf{x}_r$ since we assumed that $V_r(\mathbf{x}_r) = \mathbf{c}_{r,i'}^\top \mathbf{x}_r$. This shows that C_r does not belong to $\hat{P}$. □

Termination of Iterations: Thus, the node N_r has children of the form $\mathsf{N}_{r,\mu,i}$, each with an associated polyhedron $\mathsf{P}_{r,\mu,i}$. We will now prove convergence for the special case when the affine term in each mode is zero, i.e., for each $q \in Q$ and $j \in [l]$, $\mathbf{b}_{q,j} = \mathbf{0}$. Therefore, each dynamics are of the form $\dot{\mathbf{x}} = A_{q,j}\mathbf{x}$. Recall that for a matrix A, its spectral norm $\|A\|$ is its largest singular value, also defined as $\max_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}$.

Let $M = \max_{q\in Q} \max_{j\in[l]} \|A_{q,j}\|$ be the largest spectral norm of any matrix. Let $L > \max(2, M)$. We establish the following key lemma that shows that not only does C_r not belong to $\hat{P}_{r,\mu,i}$, all solutions within a distance of $\frac{\alpha}{L}$ are excluded as well.

Lemma 4.4. *Let $C : \langle \mathbf{c}_1, \ldots, \mathbf{c}_k \rangle$ be such that $\max_{i'\in[k]}, \|\mathbf{c}_{i'} - \mathbf{c}_{i'}^{(r)}\| \leq \frac{\alpha}{L}$. It holds that C does not belong to $\hat{P}_{r,\mu,i}$ for any μ, i.*

Proof. We will show that if C satisfies the constraints of $\hat{P}_{r,\mu,i}$, then C_r satisfies the same constraints but with α replaced by a smaller value $\alpha' > 0$. However, Lemma 4.3 can be applied to rule out this possibility.

Let us assume, for the sake of a contradiction, that C satisfies the constraints of $\hat{P}_{r,\mu,i}$ defined in (10). For all $(i, i') \in [k]^2$ with $i \neq i'$, we have

$$\begin{aligned} (\mathbf{c}_{r,i}^\top - \mathbf{c}_{i',r}^\top)\mathbf{x}_r &= (\mathbf{c}_i^\top - \mathbf{c}_{i'}^\top)\mathbf{x}_r + (\mathbf{c}_{r,i}^\top - \mathbf{c}_i^\top)\mathbf{x}_r - (\mathbf{c}_{r,i'}^\top - \mathbf{c}_{i'}^\top)\mathbf{x}_r \\ &\geq \alpha\|\mathbf{x}_r\| - \|\mathbf{c}_{r,i} - \mathbf{c}_i\|\|\mathbf{x}_r\| - \|\mathbf{c}_{r,i'} - \mathbf{c}_{i'}\|\|\mathbf{x}_r\| \\ &\geq \alpha\|\mathbf{x}_r\| - 2\frac{\alpha}{L}\|\mathbf{x}_r\| \\ &\geq \left(1 - \frac{2}{L}\right)\alpha\|\mathbf{x}_r\| \end{aligned}$$

A similar argument establishes that $\mathbf{c}_{r,i}^\top \mathbf{x}_r \geq \left(1 - \frac{1}{L}\right)\alpha\|\mathbf{x}_r\|$.

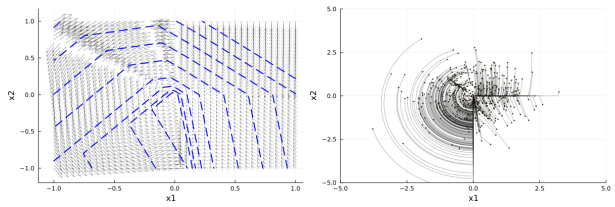


Figure 3: Vector field and trajectories of the closed loop system shown for the CLF $V(x_1, x_2)$ from Example 4.6.

Finally, let $q \in Q$ and denote $j = \mu(q)$. We have that $\mathbf{c}_i^\top (A_{q,j}\mathbf{x}_r) \leq -\alpha\|\mathbf{x}_r\|$ since we assumed $\mathbf{b}_{q,j} = \mathbf{0}$. Hence,

$$\begin{aligned}\mathbf{c}_{r,i}^\top (A_{q,j}\mathbf{x}_r) &= \mathbf{c}_i^\top A_{q,j}\mathbf{x}_r + (\mathbf{c}_{r,i}^\top - \mathbf{c}_i^\top)A_{q,j}\mathbf{x}_r \\ &\leq -\alpha\|\mathbf{x}_r\| + \|\mathbf{c}_{r,i} - \mathbf{c}_i\|\|A_{q,j}\|\|\mathbf{x}_r\| \\ &\leq -\alpha\|\mathbf{x}_r\| + \frac{\alpha}{L}M\|\mathbf{x}_r\| \\ &\leq -\left(1 - \frac{M}{L}\right)\alpha\|\mathbf{x}_r\|.\end{aligned}$$

Since $L > \max(2, M)$, we have $\frac{2}{L} < 1$ and $\frac{M}{L} < 1$. Hence, we can choose $\alpha' > 0$ such that $\alpha' \leq \min\left\{\left(1 - \frac{2}{L}\right)\alpha, \left(1 - \frac{M}{L}\right)\alpha\right\}$. It follows that C_r satisfies the constraints of $\hat{P}_{r,\mu,i}$ with α replaced by α'. Applying Lemma 4.3 yields the contradiction. □

Lemma 4.4 shows that $\hat{P}_{r,\mu,i}$ excludes C_r and a ball of radius α/L around it for some constant $L > 0$. Therefore, the depth of the tree cannot exceed $\text{pack}(\psi_0, \alpha/2L)^k$, wherein $\psi_0 : -K\mathbf{1} \leq \mathbf{c} \leq K\mathbf{1}$, and $\text{pack}(P, \delta)$ denotes the maximum number of balls of size $\delta > 0$ that can fit inside P.

Theorem 4.5. *The tree-based search algorithm terminates provided the dynamics in each mode all have* $\mathbf{b}_{q,j} = \mathbf{0}$.

Our future work will consider removing the assumption $\mathbf{b}_{q,j} = \mathbf{0}$. It requires modifying our choice of the witness $\mathbf{x}_r$ to establish bounds on its norm.

Bounds using a cutting-plane argument: In practice, the bounds provided by the closeness of distances are too large to be useful. We may obtain termination by choosing C_r to be a suitable center of the polyhedron P_r and cutting off our search whenever the volume of P_r is too small. Using well-known ideas from cutting plane methods, we can bound the depth of the tree to be a polynomial in n, k. We refer the reader to Berger et al. [6] for this approach.

Example 4.6. Returning to Example 2.3, we use the tree-based search algorithm to find a CLF with 6 pieces. The search tree was explored to a depth of 7 and discovered the CLF

$$V(x_1, x_2) = \max\left\{\begin{array}{ll} 9.86x_1 + 9.74x_2, & -1.9x_1 + 8.19x_2, \\ 9.34x_1 + 9.86x_2, & 9.87x_1 + 1.19x_2, \\ -0.95x_1 - 0.34x_2, & -5.96x_1 + 4.4x_2 \end{array}\right.$$

The overall search required 13.6 seconds. The vector field for the feedback law and the closed loop trajectories are shown in Figure 3.

5 Implementation and Empirical Evaluation

We first describe our implementation of the ideas in this paper. Our implementation uses the Julia programming language with the JuMP optimization library interface to the Gurobi optimization package (obtained through an academic license). We implemented two algorithms for searching for CLFs given a SACS: (a) the direct approach using MILP solvers; and (b) the tree search method. For the latter, we implemented two variants: one approach that systematically explored the tree up to some fixed depth using depth-first or breadth-first search and another approach based on randomized search that will be described below. We report on the performance on a few interesting benchmark systems and a set of mini-benchmarks synthesized by first fixing a quadratic CLF and designing the dynamics so that the CLF holds true. Note that despite having a quadratic CLF, our approach is not necessarily guaranteed to find a polyhedral CLF.

Implementation of Tree Search: The tree search method requires the exploration of a large search tree, wherein each node in the tree can have $O(k \times l^{|Q|})$ children. We have implemented three exploration strategies including depth-first exploration (DFS), breadth-first exploration (BFS) and a randomized strategy (RS). Description of the BFS and DFS strategies are available from an elementary algorithms textbook [11].

We will provide a brief description of the randomized search strategy. In this strategy, we are given a depth cutoff D. We explore the tree starting from the root and at each node, we pick a branch uniformly at random and explore the child node along this branch. When the depth cutoff D is reached without obtaining a solution, we backtrack to one of the ancestors of the current node, chosen uniformly at random from the ancestors. We continue the search from the chosen ancestor. Randomized search continues until some *a priori* fixed number of nodes N have been explored, and/or a timeout T has been exceeded.

However, the randomized search approach requires an important modification to be effective. Rather than pick a child node at random at each step, we pick from among all the children nodes $N_{r,\mu,i}$ whose polyhedron $P_{r,\mu,i}$ is non-empty.

Implementation "Tricks": We implemented several tricks to speedup the overall search. Here we describe the most significant ones: (a) avoiding the need for non-negativity checks by enforcing that $V(\mathbf{x})$ is positive definite by construction; and (b) breaking symmetry through a seed point.

Enforcing PSD:. First, we consider CLFs with $k + 2n$ pieces of the form $V(\mathbf{x}) = \max(\mathbf{c}_1^\top\mathbf{x}, \ldots, \mathbf{c}_k^\top\mathbf{x}, \mathbf{x}_1, -\mathbf{x}_1, \ldots, \mathbf{x}_n, -\mathbf{x}_n)$. These are automatically guaranteed to be positive definite: $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$, since $V(\mathbf{x}) \geq \|\mathbf{x}\|_\infty$. At the cost of increasing the complexity of the search, it avoids the need to check at each step that $V(\mathbf{x})$ is positive definite. In our experience, we found that this trick did not benefit the "direct approach" of section 4.1 in terms of being able to find CLFs for new problems, but had a marked benefit for the tree search method of section 4.2. Therefore, we implement this for the tree search method but not for the direct approach.

Symmetry Breaking: Next, we note that for any CLF $V(\mathbf{x}) = \max(\mathbf{c}_1^\top\mathbf{x}, \ldots, \mathbf{c}_k^\top\mathbf{x})$, and for any permutation $\pi : [k] \to [k]$, the

function $V(\mathbf{x}) = \max(\mathbf{c}_{\pi(1)}^\top \mathbf{x}, \ldots, \mathbf{c}_{\pi(k)}^\top \mathbf{x})$ is also a solution that represents the same function. A symmetry breaking constraint tries to force the search to discard such equivalent representations of the same function. Let $\mathbf{x}_0 \neq \mathbf{0}$ be an arbitrary "seed" point in the state-space of the system. We add the constraints $\mathbf{c}_1^\top \mathbf{x}_0 \geq \alpha \|\mathbf{x}_0\|$ for the fixed robustness parameter $\alpha > 0$ (discussed previously). Furthermore, we impose an ordering

$$\mathbf{c}_1^\top \mathbf{x}_0 \geq \mathbf{c}_2^\top \mathbf{x}_0 \cdots \geq \mathbf{c}_k^\top \mathbf{x}_0 \,.$$

Note that such an ordering does not lose any solutions since the coefficients $(\mathbf{c}_1, \ldots, \mathbf{c}_k)$ for any CLF can be made to satisfy it upto a permutation of the indices $1, \ldots, k$. In our experience, the addition of such a seed point benefits both search methods: the direct method (section 4.1) and the tree-search approaches 4.2. We set the seed point to $\mathbf{x}_0 = (1, \ldots, 1)$ for all our experiements.

5.1 Empirical Evaluation

We will first present empirical evaluation on some ad-hoc and simple 2D/3D examples.

Simple 2D Example: Consider a SACS with a single mode $Q = \{q\}$ over $\mathbb{R}^2$. We have $\mathcal{I}(q) = \mathbb{R}^2$ and $\mathcal{D}(q)$ has four linear but unstable dynamics of the form $\dot{\mathbf{x}} = A_i \mathbf{x}$, wherein

$$A_1 = \begin{pmatrix} -0.2 & 1 \\ 1 & 0.2 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0.5 & -1.5 \\ 0.2 & -0.1 \end{pmatrix}$$
$$A_3 = \begin{pmatrix} 0.1 & -0.1 \\ -0.1 & -0.1 \end{pmatrix} \quad A_4 = \begin{pmatrix} -1.0 & -0.1 \\ -1.0 & -0.1 \end{pmatrix}$$

Can we stablize this system through state-based switching feedback law? As it turns out, yes we can! The direct approach synthesizes a polyhedral CLF with 6 pieces in 2.1 seconds of computation time after examining 126 witnesses. The tree based approach succeeds in finding a different CLF with 6 pieces by searching the tree up to depth 7 in 1.8 seconds of computation time. For both approaches, we have considered $K = 10.0$. Figure 4 shows the resulting CLF along with closed loop trajectories generated by two feedback laws: (middle) the feedback law derived in Section 2.2 against (right) the minimum index feedback law shown in Eq. (3). This provides clear evidence that a careful choice a feedback function can avoid instabilities arising from the presence of rapid mode changes and numerical solver time-steps.

2D System with 4 Modes: Next we analyze a system taken from Berger et al. [6] wherein a polyhedral Lyapunov function is constructed. We add extra dynamics to each mode since our goal is to synthesize a feedback controller, changing the problem to one of controlling a system rather than verifying a given feedback law for stability. Each mode has 4 possible dynamics to choose from. The details of the model are provided in the Appendix. A CLF with 6 pieces and $K = 10.0$ was constructed in 14 seconds of computation time after examining 246 witnesses. Figure 5 shows the results.

A 3D Example: We formulated a SACS with 4 modes along the lines of our running example 2.3 but with 3 state variables and 4 modes. Each mode has 3 dynamics to choose from. The direct search approach generated a CLF with $k = 13$ pieces and $K = 10.0$ that was generated within 34.5 seconds after examining 144 witnesses.

Table 1: Performance of the direct approach on the micro-benchmarks. Each row represents the performance over 20 benchmark instances. k: number of pieces of the CLF (not forced to be PSD), #sol: number of instances that were solved within the timeout of 1 hour, # wit: number of witnesses examined by successful runs, ST: time taken for the successful runs (seconds).

n	k	# sol.	# wit. (avg./[min., max.])	ST (avg./ [min., max.])
2	5	20	8.2/[3.0, 34]	17.0/[3.0,127.8]
3	5	0	-	-
3	10	0	-	-
3	15	0	-	-
4	10	0	-	-
4	15	0	-	-

The generated CLF is also shown in the appendix as part of the supplementary materials.

Failing Examples: Finally, we attempted to find CLFs 6 other systems with $n \in \{3, 4\}$ state variables that were randomly generated by partitioning the space $\mathbb{R}^n$ into multiple modes and assigning stable dynamics to at least one of the modes. However, nothing in the construction of these systems guarantees that they are stabilizable or for that matter stabilizable using a polyhedral CLF.

5.2 Performance on Micro-benchmarks

We will now systematically generate benchmarks that are known to be stabilizable through a quadratic CLF. Each benchmark has a single mode $Q = \{q\}$ with $\mathcal{I}(q) = \mathbb{R}^n$ and $l = 2^r$ dynamics $\{(A_1, \mathbf{0}), \ldots, (A_l, \mathbf{0})\}$. We ensure that each A_j is not Hurwitz. The matrices A_j are generated as follows:

1. Generate a random positive definite $n \times n$ matrix Q which yields a candidate CLF $V(\mathbf{x}) = \mathbf{x}^\top Q \mathbf{x}$.
2. Generate r random linear expressions $\mathbf{a}_1^\top \mathbf{x}, \ldots, \mathbf{a}_r^\top \mathbf{x}$.
3. For each of the 2^r regions defined by the cone $C_s : \bigwedge_{i=1}^r s_i \mathbf{a}_i^\top \mathbf{x} \geq 0$, wherein $s_i \in \{-1, 1\}$,
   (a) Find a $n \times n$ matrix A such that $\mathbf{x} \in C_s \implies \mathbf{x}^\top (A^\top Q + QA)\mathbf{x} \leq -\epsilon \|\mathbf{x}\|^2$ for a fixed constant $\epsilon > 0$ by encoding the implication as a sum-of-squares problem [22, 27].
   (b) We constrain each entry of A to lie in the range $[-\gamma, \gamma]$ and set the objective to be $\mathrm{tr}(R * A)$ for a random $n \times n$ matrix A.
   (c) If the matrix A is Hurwitz, we reject the matrix A and retry using a different objective function. Otherwise, we add matrix A to the set of matrices for the mode q.

We generated 20 micro-benchmark examples each with $n = 2, 3, 4$ and $l = 4$.

Direct Approach: We first ran the direct approach of Section 4.1 on our microbenchmarks. Table 1 reports on the performance over benchmarks ranging from $n \in \{2, 3, 4\}$. We note that all the two dimensional examples were solved by the direct approach within 130 seconds with most of them taking less than 20 seconds. Unfortunately, the direct approach fails to solve any of the benchmarks beyond two dimensions.

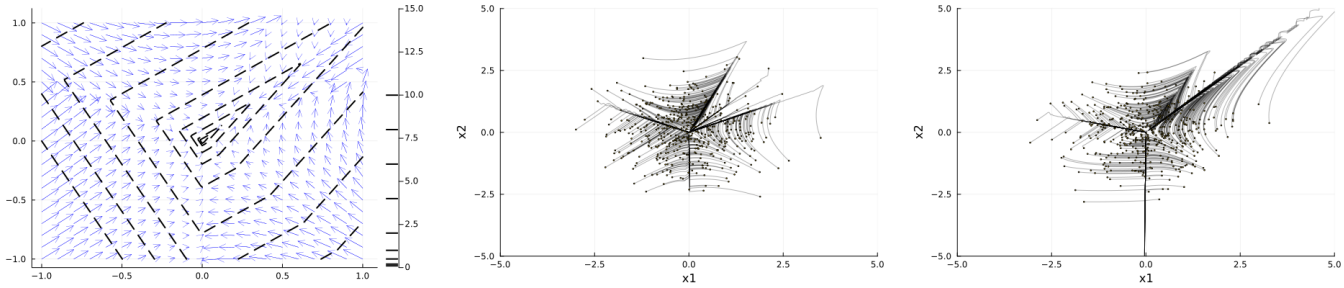


Figure 4: The polyhedral CLF (left), (mid) closed loop trajectories generated using the feedback law described in Section 2.2 contrasted with (right) the closed loop trajectories generated using the "minimum index" feedback function (Eq. (3)).

Table 2: Performance of the randomized search approach on the micro-benchmarks. Each row represents the performance over 20 benchmark instances. k: number of pieces of the CLF (forced to be PSD by adding $2n$ extra pieces), #sol: number of instances that were solved within the timeout of $T = 40$ minutes and limit on number of nodes $N = 15,000$, D: depth at which solution was found in the tree (avg/[min, max]), N: number of tree nodes examined before solution was found (avg/[min,max]) and ST: time taken for the successful runs (seconds, avg./[min,max]).

n	$k + 2n$	# sol.	D	N	ST
2	5 + 4	20	1.4 /[0,6]	26.6/[1,235]	19.4/[6.4,81.2]
3	10 + 6	17	9.25/[4,15]	1764/[21, 10606]	237.5/[2.9, 1678]
4	10 + 8	0	-	-	-
4	15 + 8	0	-	-	-

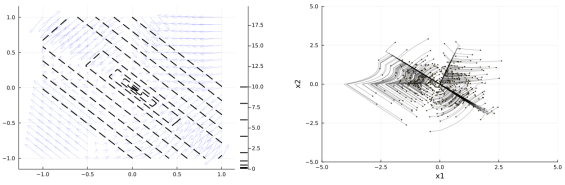


Figure 5: The polyhedral CLF (left) and closed loop trajectories generated using the feedback law described in Section 2.2.

Tree Search: We experimented with DFS and BFS-based tree search and found that randomized search dominates both approach in terms of its performance on the benchmarks for 2 and 3 dimensions. Therefore, we will omit the BFS/DFS results. Despite the fact that the search is random, it can consistently discover CLFs for these benchmarks. For our experiments, we initialized the pseudo-random number generator using a fixed seed in order to ensure reproducibility. Table 2 reports the results over the randomized tree search method. Interestingly, the approach rapidly finds CLFs for all the 2D examples and most of the 3D examples. The running time is comparable to that of the direct approach. However, we note that for $n = 4$, setting $k = 10 + 8$ or $k = 15 + 8$ is unable to yield any CLFs for a time out $T = 2400$ seconds, and a maximum limit on the number of nodes $N = 15,000$.

6 Conclusions

To conclude, we have demonstrated an approach that uses polyhedral CLFs to stabilize switched affine control systems. The key novelty lies in the characterization of feedback for polyhedral CLFs, the systematic search for a CLF using MILP solvers and demonstration over some numerical examples. The approach has the distinct advantage of not requiring a demonstrator unlike some previous approaches. At the same time, however, it is computationally expensive both in theory and practice. In the future, we hope to make our algorithm faster by combining it with ideas from machine learning that would enable us to use to rapidly learn candidate CLFs while using verification and counterexample generation to drive iterative exploration. We are also interested in extensions of our approach to nonlinear hybrid systems.

Acknowledgments

We thank the anonymous reviewer for detailed comments on this submission. This work was funded, in part, by the US National Science Foundation (NSF) under award number CNS-1836900. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. 2021. FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th international conference on hybrid systems: computation and control*. 1–11.

[2] Daniele Ahmed, Andrea Peruffo, and Alessandro Abate. 2020. Automated and sound synthesis of Lyapunov functions with SMT solvers. In *Tools and Algorithms for the Construction and Analysis of Systems: 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings, Part I 26*. Springer, 97–114.

[3] Roberto Ambrosino, Marco Ariola, and Francesco Amato. 2012. A convex condition for robust stability analysis via polyhedral Lyapunov functions. *SIAM Journal on Control and Optimization* 50, 1 (2012), 490–506.

[4] Jean-Pierre Aubin and Arrigo Cellina. 1984. *Differential inclusions: set-valued maps and viability theory*. Springer, Berlin. https://doi.org/10.1007/978-3-642-69512-4

[5] Robert Baier, Philipp Braun, Lars Grüne, and Christopher M Kellett. 2019. Numerical calculation of nonsmooth control lyapunov functions via piecewise affine approximation. *IFAC-PapersOnLine* 52, 16 (2019), 370–375.

[6] Guillaume O Berger and Sriram Sankaranarayanan. 2022. Learning fixed-complexity polyhedral Lyapunov functions from counterexamples. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 3250–3255.

[7] Guillaume O Berger and Sriram Sankaranarayanan. 2023. Counterexample-guided computation of polyhedral Lyapunov functions for piecewise linear systems. *Automatica* 155 (2023), 111165.

[8] Franco Blanchini, Stefano Miani, et al. 2008. *Set-theoretic methods in control*. Vol. 78. Springer.

[9] Ya-Chien Chang, Nima Roohi, and Sicun Gao. 2019. Neural lyapunov control. *Advances in neural information processing systems* 32 (2019).

[10] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. 2021. Learning lyapunov functions for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 1–11.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3rd ed.). MIT Press, Cambridge, MA.

[12] Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. 2020. Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 1274–1281.

[13] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. 2021. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152* (2021).

[14] Hongkai Dai and Frank Permenter. 2023. Convex synthesis and verification of control-Lyapunov and barrier functions with input constraints. In *2023 American Control Conference (ACC)*. IEEE, 4116–4123.

[15] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[16] Alec Edwards, Andrea Peruffo, and Alessandro Abate. 2024. Fossil 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. 1–10.

[17] Sicun Gao, Soonho Kong, and Edmund M Clarke. 2013. dReal: An SMT solver for nonlinear theories over the reals. In *International conference on automated deduction*. Springer, 208–214.

[18] Nicola Guglielmi, Linda Laglia, and Vladimir Protasov. 2017. Polytope Lyapunov functions for stable and for stabilizable LSS. *Foundations of Computational Mathematics* 17 (2017), 567–623.

[19] James Kapinski, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. 2014. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*. 133–142.

[20] S Mohammad Khansari-Zadeh and Aude Billard. 2014. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems* 62, 6 (2014), 752–765.

[21] Dimitris Kousoulidis and Fulvio Forni. 2021. Polyhedral Lyapunov functions with fixed complexity. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 3293–3298.

[22] Jean B Lasserre. 2001. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization* 11, 3 (2001), 796–817.

[23] Fabien Lauer and Gérard Bloch. 2018. *Hybrid system identification: Theory and algorithms for learning switching models*. Vol. 478. Springer.

[24] Mircea Lazar and Alina I Doban. 2011. On infinity norms as Lyapunov functions for continuous-time dynamical systems. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 7567–7572.

[25] Daniel Liberzon. 2003. *Switching in systems and control*. Vol. 190. Springer.

[26] Stefano Miani and Carlo Savorgnan. 2005. MAXIS-G: a software package for computing polyhedral invariant sets for constrained LPV systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 7609–7614.

[27] Pablo A Parrilo. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology.

[28] Andrzej Polański. 2000. On absolute stability analysis by polyhedral Lyapunov functions. *Automatica* 36, 4 (2000), 573–578.

[29] Hadi Ravanbakhsh and Sriram Sankaranarayanan. 2015. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *2015 54th IEEE conference on decision and control (CDC)*. IEEE, 4232–4239.

[30] Hadi Ravanbakhsh and Sriram Sankaranarayanan. 2019. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots* 43 (2019), 275–307.

[31] Eduardo D Sontag. 1983. A Lyapunov-like characterization of asymptotic controllability. *SIAM journal on control and optimization* 21, 3 (1983), 462–471.

[32] Zhendong Sun. 2006. *Switched linear systems: control and design*. Springer Science & Business Media.

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

[34] Ufuk Topcu, Andrew Packard, and Peter Seiler. 2008. Local stability analysis using simulations and sum-of-squares programming. *Automatica* 44, 10 (2008), 2669–2675.

[35] Robert J Vanderbei. 2020. *Linear programming: foundations and extensions* (5^{th} ed.). Springer, Cham. https://doi.org/10.1007/978-1-4614-7630-6