Noname manuscript No.
(will be inserted by the editor)

Learning Control Lyapunov Functions from Counterexamples and Demonstrations

Hadi Ravanbakhsh · Sriram Sankaranarayanan

Received: date / Accepted: date

Abstract We present a technique for learning control Lyapunov-like functions, which are used in turn to synthesize controllers for nonlinear dynamical systems that can stabilize the system, or satisfy specifications such as remaining inside a safe set, or eventually reaching a target set while remaining inside a safe set. The learning framework uses a *demonstrator* that implements a black-box, untrusted strategy presumed to solve the problem of interest, a *learner* that poses finitely many queries to the demonstrator to infer a candidate function, and a *verifier* that checks whether the current candidate is a valid control Lyapunov function. The overall learning framework is iterative, eliminating a set of candidates on each iteration using the counterexamples discovered by the verifier and the demonstrations over these counterexamples. We prove its convergence using ellipsoidal approximation techniques from convex optimization. We also implement this scheme using nonlinear MPC controllers to serve as demonstrators for a set of state and trajectory stabilization problems for nonlinear dynamical systems. We show how the verifier can be constructed efficiently using convex relaxations of the verification problem for polynomial systems to semi-definite programming (SDP) problem instances. Our approach is able to synthesize relatively simple polynomial control Lyapunov functions, and in that process replace the MPC using a guaranteed and computationally less expensive controller.

H. Ravanbakhsh
University of Colorado, Boulder
E-mail: hadi.ravanbakhsh@colorado.edu

S. Sankaranarayanan
University of Colorado, Boulder
E-mail: sriram.sankaranarayanan@colorado.edu

Keywords Lyapunov Functions · Controller Synthesis · Learning from Demonstrations · Concept Learning.

1 Introduction

We propose a novel *learning from demonstration* scheme for inferring control Lyapunov functions (potential functions) for stabilizing nonlinear dynamical systems to reference states/trajectories, and implementing control laws for specifications such as maintaining a system inside a set of safe states, reaching a target set while remaining inside a safe set and tracking a given trajectory while not deviating too far away. Control Lyapunov functions (CLFs) have wide applications to autonomous systems [37, 30, 3, 63, 83]. They extend the classic notion of Lyapunov functions to systems involving control inputs [87, 88, 6]. Finding a CLF also leads us to an associated feedback control law that can be used to solve the stabilization problem. Additionally, they can be extended for feedback motion planning using extensions to time-varying or sequential CLFs [19, 96]. Likewise, they have been investigated in the robotics community in many forms including *artificial potential functions* to solve path planning problems involving obstacles [55].

However, synthesizing CLFs for nonlinear systems remains a challenge [73]. Standard approaches to finding CLFs include the use of dynamic programming, wherein the value function satisfies the conditions of a CLF [12], or using non-convex bilinear matrix inequalities (BMI) [35].

In this article, we investigate the problem of learning a CLF using a black-box DEMONSTRATOR that implements an unknown state feedback law to stabilize the system to a given equilibrium. This DEMONSTRATOR can be queried at a given system state, and returns a *demonstration* in the form of a control input generated at that state by its feedback law. Such a DEMONSTRATOR can be realized using an expensive nonlinear model predictive controller (MPC) that uses a local optimization scheme, or even a human operator under certain assumptions [1]. Additionally, the framework has a LEARNER which selects a candidate CLF and a VERIFIER that tests whether this CLF is valid. If the CLF is invalid, the VERIFIER returns a state at which the current candidate fails. The LEARNER queries the DEMONSTRATOR to obtain a control input corresponding to this state. It subsequently eliminates the current candidate along with a set of related functions from further consideration. The framework continues to exhaust the space of candidate CLFs until no CLFs remain or a valid CLF is found in this process. We prove the process can converge in finitely many steps provided the LEARNER chooses the candidate function appropriately at each step. We also provide efficient SDP-based approximations to the verification problem that can be used to drive the framework. Finally, we test this approach on a variety of examples, by solving stabilization problems for nonlinear dynamical systems. We show that our approach can successfully find CLFs using finite horizon nonlinear MPC schemes with appropriately chosen cost functions to serve as demonstrators. In these instances, the CLFs yield control laws that are computationally inexpensive, and guaranteed against the original dynamical model.

This paper is an extended version of our earlier work [80]. When compared to the earlier work, we have thoroughly expanded the technical sections to provide detailed proofs of the various results and a detailed exposition of each component of our learning framework. Additionally, we have included a new section that discusses specifications other than stability properties. We have also extended our experimental results and compare different options for implementing the overall learning loop as well as comparisons with other methods. We also provide a detailed discussion of various extensions to the approach presented in this paper.

1.1 Illustrative Example: TORA System

Figure 1(a) shows a mechanical system, called translational oscillations with a rotational actuator (TORA). The system consists of a cart attached to a wall using a spring. Inside the cart, there is an arm with a weight which can rotate. The cart itself can oscillate freely and there are no friction forces. The system has two degrees of freedom, including the position of the cart x, and the rotational position of the arm θ. The controller can rotate the arm through input u. The goal is to stabilize the cart to $x = 0$, with its velocity, angle, and angular velocity $\dot{x} = \theta = \dot{\theta} = 0$. We refer the reader to Jankovic et al. [38] for a derivation of the dynamics, shown below in terms of state variables $(x_1, \ldots, x_4)$, collectively written as a vector $\mathbf{x}$, and a single control input (u_1), written as a vector $\mathbf{u}$, after a basis transformation:

$$\dot{x_1} = x_2,\ \dot{x_2} = -x_1 + \epsilon \sin(x_3),\ \dot{x_3} = x_4,\ \dot{x_4} = u_1\,. \quad (1)$$

$\sin(x_3)$ is approximated using a degree three polynomial approximation which is quite accurate over the range $x_3 \in [-2, 2]$. The equilibrium $x = \dot{x} = \theta = \dot{\theta} = 0$ now corresponds to $x_1 = x_2 = x_3 = x_4 = 0$. The system has a single control input u_1 that is bounded $u_1 \in [-1.5, 1.5]$. Further, we define a "safe set" $S : [-1, 1] \times [-1, 1] \times [-2, 2] \times [-1, 1]$, so that if $\mathbf{x}(0) \in S$ then $\mathbf{x}(t) \in S$ for all time $t \geq 0$.

MPC Scheme: A first approach to solve the problem uses a nonlinear model-predictive control (MPC) scheme using a discretization of the system dynamics with time step $\tau = 1$. The time t belongs to set $\{0, \tau, 2\tau, \ldots, N\tau = \mathcal{H}\}$ and:

$$\mathbf{x}(t + \tau) = \mathbf{x}(t) + \tau f(\mathbf{x}(t), \mathbf{u}(t))\,, \quad (2)$$

with $f(\mathbf{x}, \mathbf{u})$ representing the vector field of the ODE in (1). Fixing the time horizon $\mathcal{H} = 30$, we use a simple cost function $J(\mathbf{x}(0), \mathbf{u}(0), \mathbf{u}(\tau), \ldots, \mathbf{u}(\mathcal{H} - \tau)\})$:

$$\sum_{t \in \{0, \tau, \ldots, \mathcal{H} - \tau\}} \left(||\mathbf{x}(t)||_2^2 + ||\mathbf{u}(t)||_2^2\right) + N\ ||\mathbf{x}(\mathcal{H})||_2^2\,. \quad (3)$$

Here, we constrain $\mathbf{u}(t) \in [-1.5, 1.5]$ for all t and define $\mathbf{x}(t + \tau)$ in terms of $\mathbf{x}(t)$ using the discretization in (2). Such a control is implemented using a first/second order numerical gradient descent method to minimize the cost function [64]. The stabilization of the system was informally confirmed through hundreds of simulations from different initial states. However, the MPC scheme is expensive, requiring repeated solutions to (constrained) nonlinear optimization problems in real-time. Furthermore, in general, the closed loop lacks formal guarantees despite the *high confidence* gained from numerous simulations.

Learning a Control Lyapunov Function: In this article, we introduce an approach which uses the MPC scheme as a DEMONSTRATOR, and attempts to learn a control Lyapunov function. Then, a control law (in a closed form) is obtained from the CLF. The overall idea, depicted in Fig. 2, is to pose queries to the *offline* MPC at finitely many *witness* states $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(j)}\}$. Then,

[1] However, we do not handle noisy or erroneous demonstrators in this paper.

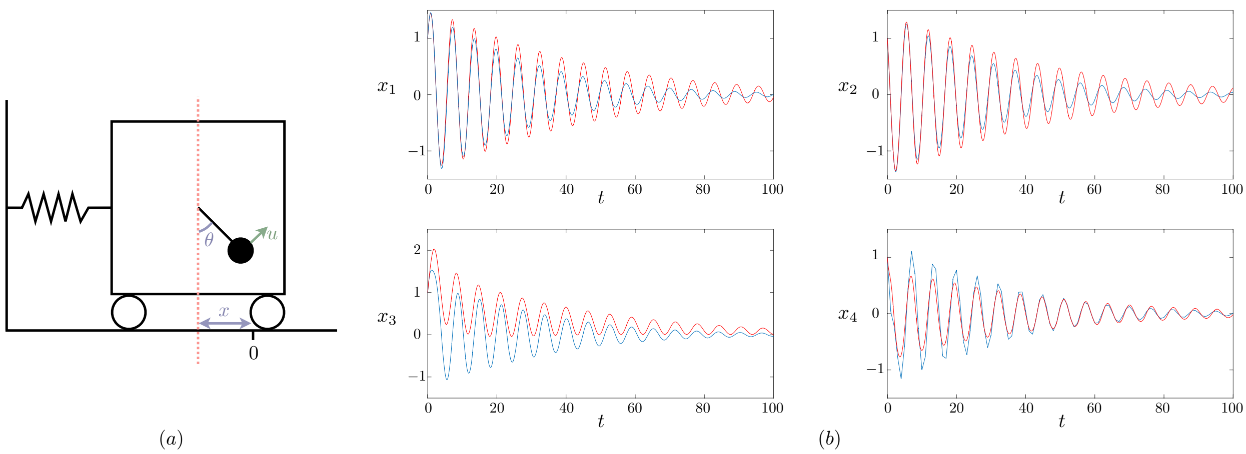


Fig. 1 TORA System. (a) A schematic diagram of the TORA system. (b) Execution traces of the system using MPC control (blue traces) and Lyapunov based control (red traces) starting from same initial point.

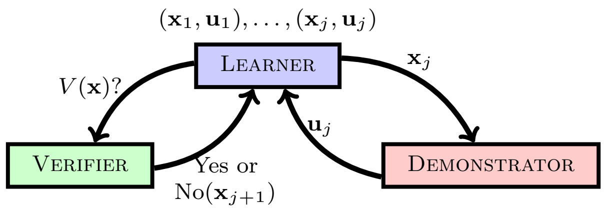


Fig. 2 Overview of the learning framework for learning a control Lyapunov function.

for each witness state $\mathbf{x}^{(i)}$, the MPC is applied to generate a sequence of control inputs $\mathbf{u}^{(i)}(0), \mathbf{u}^{(i)}(\tau), \cdots, \mathbf{u}^{(i)}(\mathcal{H}-\tau)$ with $\mathbf{x}^{(i)}$ as the initial state, in order to drive the system into the equilibrium starting from $\mathbf{x}^{(i)}$. The MPC then retains the first control input $\mathbf{u}^{(i)} :\ \mathbf{u}^{(i)}(0)$, and discards the remaining (as is standard in MPC). This yields the so called observation pairs $(\mathbf{x}^{(i)}, \mathbf{u}^{(i)})$ that are used by the LEARNER.

The LEARNER attempts to find a candidate function $V(\mathbf{x})$ that is positive definite and which decreases at each witness state $\mathbf{x}^{(i)}$ through the control input $\mathbf{u}^{(i)}$. This function V is potentially a CLF function for the system. This function is fed to the VERIFIER, which checks whether $V(\mathbf{x})$ is indeed a CLF, or discovers a state $\mathbf{x}^{(j+1)}$ which refutes V. This new state is added to the witness set and the process is iterated. The procedure described in this paper synthesizes the control Lyapunov function $V(\mathbf{x})$ below:

$$\begin{aligned} V =& 1.22x_2^2 + 0.31x_2x_3 + 0.44x_3^2 - 0.28x_4x_2 \\ &+ 0.80x_4x_3 + 1.69x_4^2 + 0.07x_1x_2 - 0.66x_1x_3 \\ &- 1.85x_4x_1 + 1.6x_1^2 \,. \end{aligned}$$

Next, this function is used to design a associated control law that guarantees the stabilization of the model described in Eq. (1). Figure 1(b) shows a closed loop trajectory for this control law vs control law extracted by the MPC. At each step, given a current state $\mathbf{x}$, we compute an input $\mathbf{u} \in [-1.5, 1.5]$ such that:

$$(\nabla V) \cdot f(\mathbf{x}, \mathbf{u}) < 0 \,. \tag{4}$$

First, the definition of a CLF guarantees that any state $\mathbf{x} \in S$, a control input $\mathbf{u} \in [-1.5, 1.5]$ that satisfies Eq. (4) exists. Such a $\mathbf{u}$ may be chosen directly by means of a formula involving $\mathbf{x}$ [53,91] unlike the MPC which solves a nonlinear problem in Eq. (3). Furthermore, the resulting control law guarantees the stability of the resulting closed loop.

2 Background

We recall preliminary notions, including the stabilization problem for nonlinear dynamical systems.

2.1 Problem Statement

We will first define the system model studied throughout this paper.

Definition 1 (Control System) A state feedback control system $\Psi(X, U, f, \mathcal{K})$ consists of a plant, a controller over $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$.

1. $X \subseteq \mathbb{R}^n$ is the *state space* of the system. The control inputs belong to a set U defined as a polyhedron:

$$U = \{\mathbf{u} \mid A\mathbf{u} \geq \mathbf{b}\} \,. \tag{5}$$

2. The plant consists of a vector field defined by a continuous and differentiable function $f : X \times U \mapsto \mathbb{R}^n$.

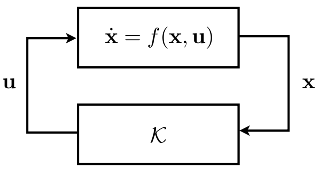


Fig. 3 Closed-loop state feedback system.

3. The controller measures the state of the plant $\mathbf{x} \in X$ and provides feedback $\mathbf{u} \in U$. The controller is defined by a feedback function $\mathcal{K} : X \mapsto U$ (Fig. 3).

For now, we assume $\mathcal{K}$ is a smooth (continuous and differentiable) function. For a given feedback law $\mathcal{K}$, an execution trace of the system, starting from an initial state $\mathbf{x}_0$ is a function: $\mathbf{x} : [0, T(\mathbf{x}_0)) \mapsto X$, which maps time $t \in [0, T(\mathbf{x}_0))$ to a state $\mathbf{x}(t)$, such that

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathcal{K}(\mathbf{x}(t)))\,,$$

where $\dot{\mathbf{x}}(\cdot)$ is the right derivative of $\mathbf{x}(\cdot)$ w.r.t. time over $[0, T(\mathbf{x}_0))$. Since f and $\mathcal{K}$ are assumed to be smooth, there exists a unique trajectory for any $\mathbf{x}_0$, defined over some time interval $[0, T(\mathbf{x}_0))$. Here $T(\mathbf{x}_0)$ is ∞ if trajectory starting from $\mathbf{x}_0$ exists for all time. Otherwise, $T(\mathbf{x}_0)$ is finite if the trajectory "escapes" in finite time. For most of the systems we study, the closed loop dynamics are such that a compact set S will be positive invariant. In fact, this set will be a sublevel set of a Lyapunov function for the closed loop dynamics. This fact along with the smoothness of $f, \mathcal{K}$ suffices to establish that $T(\mathbf{x}_0) = \infty$ for all $\mathbf{x}_0 \in S$. Unless otherwise noted, we will consider control laws $\mathcal{K}$ that will guarantee existence of trajectories for all time.

A specification describes the desired behavior of all possible execution traces $\mathbf{x}(\cdot)$. In this article, we study a variety of specifications, including stability, trajectory tracking, and safety. For simplicity, we first focus on stability. Extensions to other specifications are presented in Section 6. Also, without loss of generality, we assume $\mathbf{x} = \mathbf{0}$ is the desired equilibrium. Moreover, $f(\mathbf{0}, \mathbf{0}) = \mathbf{0}$.

Problem 1 (Synthesis for Asymptotic Stability) Given a plant, the control synthesis problem is to design a controller (a feedback law $\mathcal{K}$) s.t. all traces $\mathbf{x}(\cdot)$ of the closed loop system $\Psi(X, U, f, \mathcal{K})$ are asymptotically stable. We require two properties for asymptotic stability. First, the system is *Lyapunov stable*:

$$\begin{array}{l}(\forall \epsilon > 0)\\ \quad (\exists \delta > 0)\\ \qquad \left(\begin{array}{c}\forall \mathbf{x}(\cdot)\\ \mathbf{x}(0) \in B_\delta(\mathbf{0})\end{array}\right)\ (\forall t \geq 0)\ \mathbf{x}(t) \in \mathcal{B}_\epsilon(\mathbf{0})\,,\end{array}$$

wherein $\mathcal{B}_\delta(\mathbf{x}) \subseteq \mathbb{R}^n$ is the ball of radius δ centered at $\mathbf{x}$. In other words, for any chosen $\epsilon > 0$, we may ensure that the trajectories will stay inside a ball of ϵ radius by choosing the initial conditions to lie inside a ball of δ radius.

Furthermore, all the trajectories converge asymptotically towards the origin:

$$(\forall \epsilon > 0)\ (\forall \mathbf{x}(\cdot))\ (\exists T > 0)\ (\forall t \geq T)\ \mathbf{x}(t) \in \mathcal{B}_\epsilon(\mathbf{0})\,.$$

I.e., For any chosen $\epsilon > 0$, all trajectories will eventually reach a ball of radius ϵ around the origin and stay inside forever.

Stability in our method is addressed through Lyapunov analysis. More specifically, our solution is based on control Lyapunov functions (CLF). CLFs were first introduced by Sontag [87,88], and studied at the same time by Artstein [6]. Sontag's work shows that if a system is asymptotically stablizable, then there exists a CLF even if the dynamics are not smooth [88]. Now, let us recall the definition of a positive and negative definite functions.

Definition 2 (Positive Definite) A function $V : \mathbb{R}^n \mapsto \mathbb{R}$ is *positive definite* over a set X containing $\mathbf{0}$, iff $V(\mathbf{0}) = 0$ and $V(\mathbf{x}) > 0$ for all $\mathbf{x} \in X \setminus \{\mathbf{0}\}$.

Likewise, V is *negative definite* iff $-V$ is positive definite.

Definition 3 (Control Lyapunov Function(CLF)) A smooth, radially unbounded function V is a control Lyapunov function (CLF) over X, if the following conditions hold [6]:

$$\begin{array}{l}V \text{ is positive definite over } X\\ \min_{\mathbf{u} \in U} (\nabla V) \cdot f(\mathbf{x}, \mathbf{u}) \text{ is negative definite over} X\,,\end{array} \tag{6}$$

where ∇V is the gradient of V. Note that $(\nabla V) \cdot f$ is the Lie derivative of V according to the vector field f.

Another way of interpreting the second condition is that for each $\mathbf{x} \in X$, a control $\mathbf{u} \in U$ can be chosen to ensure an *instantaneous decrease* in the value of V, as illustrated in Fig. 4.

Solving Stabilization using CLFs: Finding a CLF V guarantees the existence of a feedback law that can stabilize all trajectories to the equilibrium [6]. However, constructing such a feedback law is not trivial and potentially expensive. Further results can be obtained by restricting the vector field f to be control affine:

$$f(\mathbf{x}, \mathbf{u}) :\ f_0(\mathbf{x}) + \sum_{i=1}^{m} f_i(\mathbf{x}) u_i\,, \tag{7}$$

wherein $f_i : X \mapsto \mathbb{R}[X]^n$. Assuming $U : \mathbb{R}^m$, Sontag provides a method for extracting a feedback law $\mathcal{K}$, for

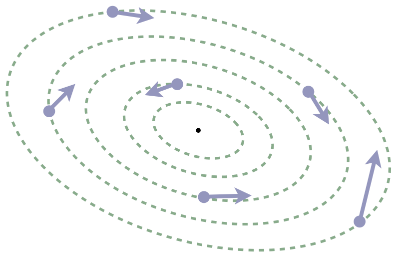


Fig. 4 Control Lyapunov Function (CLF): Level-sets of a CLF V are shown using the green lines. For each state (blue dot), the vector field $f(\mathbf{x}, \mathbf{u})$ for $\mathbf{u} = \mathcal{K}(\mathbf{x})$ is the blue arrow, and it points to a direction which decreases V.

control affine systems from a control Lyapunov function [89]. More specifically, if a CLF V is available, the following feedback law stabilizes the system:

$$\mathcal{K}_i(\mathbf{x}) = \begin{cases} 0 & \beta(\mathbf{x}) = 0 \\ -b_i(\mathbf{x}) \frac{a(\mathbf{x}) + \sqrt{a(\mathbf{x})^2 + \beta(\mathbf{x})^2}}{\beta(\mathbf{x})} & \beta(\mathbf{x}) \neq 0 \,, \end{cases} \tag{8}$$

where $a(\mathbf{x}) = \nabla V.f_0(\mathbf{x})$, $b_i(\mathbf{x}) = \nabla V.f_i(\mathbf{x})$, and $\beta(\mathbf{x}) = \sum_{i=1}^m b_i^2(\mathbf{x})$.

Remark 1 Feedback law $\mathcal{K}$ provided by the Sontag formula is not necessarily continuous at the origin. Nevertheless, such a feedback law still guarantees stabilization. See [89] for more details.

Sontag formula can be extended to systems with saturated inputs where U is an n-ball [53] or a polytope [91]. Also switching-based feedback is possible, under some mild assumptions (to avoid Zeno behavior) [22,77]. We assume dynamics are affine in control and use these results which reduce Problem 1 to that of finding a control Lyapunov function V.

2.2 Discovering CLFs

We briefly summarize approaches for discovering CLFs for a given plant model in order to stabilize it to a given equilibrium state. Efficient methods for discovering CLFs are available only for specific classes of systems such as feedback linearizable systems, or for so-called strict feedback systems, wherein a procedure called *backstepping* can be used [28]. However, finding CLFs for general nonlinear systems is challenging [73].

One class of solutions uses optimal control theory by setting up the problem of stabilization as one of minimizing a cost function over the trajectories of the system. If the cost function is set up appropriately, then the value function for the resulting dynamic programming problem is a a CLF [73,12]. To do so, however, one needs to solve a Hamilton-Jacobi-Bellman (HJB) partial differential equation to discover the value function, which can be quite hard in practice[18]. In fact, rather than solve HJB equations to obtain CLFs, it is more common to derive a CLF using a procedure such as backstepping and apply inverse optimality results to derive cost functions [28].

A second class of solution is based on parameterization. More specifically, a class of function $V_{\mathbf{c}}(\mathbf{x})$ is parameterized by a set of unknown parameters $\mathbf{c}$. This parameterization is commonly specified as a linear combination of basis functions of the form $V_{\mathbf{c}}(\mathbf{x}) : \sum c_i g_i(\mathbf{x})$. Furthermore, the functions g_i commonly range over all possible monomials up to some prespecified degree limit D. Next, an instantiation of the parameters $\mathbf{c}$ is discovered so that the resulting function V is a CLF. Unfortunately, discovering such parameters requires the solution to a quantifier elimination problem, in general. This is quite computationally expensive for nonlinear systems. Previously, authors proposed a framework which uses sampling to avoiding expensive quantifier eliminations [78]. Despite the use of sampling, scalability remains an issue. Another solution is based on sum-of-squares relaxations [84,47,66], along the lines of approaches used to discover Lyapunov functions [65]. However, discovering CLFs using this approach entails solving a system of bilinear matrix inequalities [94,35]. In contrast to LMIs, the set of solutions to a BMIs form a nonconvex set, and solving BMIs is well-known to be computationally expensive, in practice. Rather than solving a BMI to find a CLF, and then extracting the feedback law from the CLF, an alternative approach is to simultaneously search for a Lyapunov function V and an unknown feedback law at the same time [25,94,56]. The latter approach also yields bilinear matrix inequalities of comparable sizes. Rather than seek algorithms that are guaranteed to solve BMIs, a simpler approach is to attempt to solve the BMIs using *alternating minimization*: a form of coordinate descent that fixes one set of variables in BMI, obtaining an LMI over the remaining variables. However, these approaches usually stuck in a local "saddle point", and fail as a result [33].

Approaches that parameterize a family of functions $V_{\mathbf{c}}(\mathbf{x})$ face the issue of choosing a family such that a CLF belonging to that family is known to exist whenever the system is asymptotically stabilizable in the first place. There is a rich literature on the existence of CLFs for a given class of plant models. As mentioned earlier, if a system is *asymptotically stablizable*, then there exists a CLF even if the dynamics are not smooth [88]. However, the CLF does not have to be smooth. Recent results, have shown some light on the existence of polynomial Lyapunov functions for certain classes of systems. Peet showed that an exponentially stable system has a polynomial local Lyapunov function over a bounded region [67]. Thus, if there exists some feedback law that exponentially stabilizes a given plant, we may conclude the existence of a polynomial CLF for that system. This was recently extended to rationally stable systems i.e., the distance to equilibrium decays as $o(t^{-k})$ for trajectories starting from some set Ω, by Leth et al. [51]. These results do not guarantee that a search for a polynomial CLF will be successful due to the lack of a bound on the degree D. This can be addressed by increasing the degree of the monomials until a CLF is found, but the process can be prohibitively expensive. Another drawback is that most approaches use SOS relaxations over polynomial systems to check the CLF conditions, although there is no guarantee as yet that polynomial CLFs that are also verifiable through SOS relaxations exist.

Another class of solutions involves approximate dynamic programming to find approximations to value functions [13]. The solutions obtained through these approaches are not guaranteed to be CLFs and thus may need to be discarded, if the final result does not satisfy the conditions for a CLF. Approximate solutions are also investigated through learning from demonstrations [106]. Khansari-Zadeh et al. learn a CLF from demonstrations through a combination of sampling states and corresponding feedback provided by the demonstrator. A likely CLF is learned through parameterizing a class of functions $V_{\mathbf{c}}(\mathbf{x})$, and finding conditions on $\mathbf{c}$ by enforcing the conditions for the CLFs at the sampled states [83]. The conditions for being a CLF should be checked on the solution obtained by solving these constraints.

Compared to the techniques described above, the approach presented in this paper is based on parameterization by choosing a class of functions $V_{\mathbf{c}}(\mathbf{x})$ and attempting to find a suitable $\mathbf{c} \in C$ so that the result is a CLF. Our approach avoids having to solve BMIs by instead choosing finitely many sample states, and using demonstrator's feedback to provide corresponding sample controls for the state samples. However, instead of choosing these samples at random, we use a verifier to select samples. Furthermore, our approach can also systematically explore the space of possible parameters C in a manner that guarantees termination in number of iterations polynomial in the dimensionality of C and $\mathbf{x}$. The result upon termination can be a guaranteed CLF V or failure to find a CLF among the class of functions provided.

3 Formal Learning Framework

As mentioned earlier, finding a control Lyapunov function is computationally expensive, requiring the solution to BMIs [94] or hard non-linear constraints [77]. The goal is to search for a solution (CLF) over a hypothesis space. More specifically, a CLF is parameterized by a set of unknown parameters $\mathbf{c} \in C$ ($C \subseteq \mathbb{R}^r$). The parameterized CLF is shown by $V_{\mathbf{c}}$. And the goal is to find $\mathbf{c} \in C$ s.t.

$$\begin{array}{l} V_{\mathbf{c}} \text{ is positive definite} \\ \min_{\mathbf{u}\in U} \nabla V_{\mathbf{c}} \cdot f(\mathbf{x},\mathbf{u}) \text{ is negative definite}\,. \end{array} \tag{9}$$

A standard approach is to choose a set of basis functions $g_1, \ldots, g_r$ ($g_i : X \mapsto \mathbb{R}$) and search for a function of the form

$$V_{\mathbf{c}}(\mathbf{x}) = \sum_{j=1}^{r} c_j g_j(\mathbf{x})\,. \tag{10}$$

Remark 2 The basis functions are chosen s.t. $V_{\mathbf{c}}$ is radially unbounded and smooth, independent of the coefficients.

As mentioned earlier, the learning framework has three components: a demonstrator, a learner, and a verifier (see Fig. 2). The demonstrator inputs a state $\mathbf{x}$ and returns a control input $\mathbf{u} \in U$, that is an appropriate "instantaneous" feedback for $\mathbf{x}$. Formally, demonstrator is a function $\mathcal{D} : X \mapsto U$.

Remark 3 (Demonstrator) The demonstrator is treated as a black box. This allows to use a variety of approaches ranging from trajectory optimization [105], human expert demonstrations [83], and sample-based methods [50,45], which can be probabilistically complete. While the demonstrator is *presumed* to stabilize the system, our method can work even if the demonstrator is faulty. Specifically, a faulty demonstrator in worst case, may cause our method to terminate without having found a CLF. However, if a CLF is found by our approach, it is guaranteed to be correct.

The formal learning procedure receives inputs:

1. A plant described by f
2. A "black-box" demonstrator function $\mathcal{D} : X \mapsto U$
3. A set of basis functions $g_1, \ldots, g_r$ to form the hypothesis space $V_{\mathbf{c}}(\mathbf{x}) : \sum_{j=1}^{r} c_j g_j(\mathbf{x})$,

and either (a) outputs a $\mathbf{c} \in C$ s.t. $V_{\mathbf{c}}(\mathbf{x}) : \mathbf{c}^t \cdot \mathbf{g}(\mathbf{x})$ is a CLF (Eq. (9)); or (b) declares FAILURE: no CLF could be discovered.

The goal of this framework is to find a CLF from a finite set of queries to a demonstrator.

Definition 4 (Observations) We define a set of observations O as

$$O : \{(\mathbf{x}_1, \mathbf{u}_1), \ldots, (\mathbf{x}_j, \mathbf{u}_j)\} \subset X \times U\,,$$

where $\mathbf{u}_i$ is the demonstrated feedback for state $\mathbf{x}_i$, i.e., $\mathbf{u}_i : \mathcal{D}(\mathbf{x}_i)$. Further, we will assume that $\mathbf{x}_i \neq \mathbf{0}$.

Definition 5 (Observation Compatibility) A function V is said to be compatible with a set of observations O iff V respects the CLF conditions (Eq. (6)) for every observation in O:

$$V(\mathbf{0}) = 0 \;\wedge \bigwedge_{(\mathbf{x}_i,\mathbf{u}_i)\in O_j} \begin{pmatrix} V(\mathbf{x}_i) > 0 \;\wedge \\ \nabla V \cdot f(\mathbf{x}_i, \mathbf{u}_i) < 0 \end{pmatrix}.$$

We note that observation compatible functions need not necessarily be a CLF, since they may violate the CLF condition for some state $\mathbf{x}$ that is not part of an observation in O. On the flip side, not every CLF (satisfying the conditions in Eq. (6)) will necessarily be compatible with a given observation set O.

Definition 6 (Demonstrator Compatibility) A function V is said to be compatible with a demonstrator $\mathcal{D}$ iff V respects the CLF conditions (Eq. (6)) for every observation that can be generated by the demonstrator:

$$V(\mathbf{0}) = 0 \;\wedge\; \forall \mathbf{x} \neq \mathbf{0} \begin{pmatrix} V(\mathbf{x}) > 0 \;\wedge \\ \nabla V \cdot f(\mathbf{x}, \mathcal{D}(\mathbf{x})) < 0 \end{pmatrix}.$$

In other words, V is a Lyapunov function for the closed loop system $\Psi(X, U, f, \mathcal{D})$.

Now, we describe the learning framework. The framework consists of a learner and a verifier. The learner interacts with the verifier and the demonstrator. The framework works iteratively and at each iteration j the learner maintains a set of observations

$$O_j : \{(\mathbf{x}_1, \mathbf{u}_1), \ldots, (\mathbf{x}_j, \mathbf{u}_j)\} \subset X \times U\,.$$

Corresponding to O_j, $C_j \subseteq C$ is defined as a set of candidate unknowns for function $V_{\mathbf{c}}(\mathbf{x})$. Formally, C_j is a set of all $\mathbf{c}$ s.t. $V_{\mathbf{c}}$ is compatible with O_j:

$$C_j : \left\{ \mathbf{c} \in C \;\middle|\; \begin{array}{c} V_{\mathbf{c}}(0) = 0 \;\wedge \\ \bigwedge_{(\mathbf{x}_i,\mathbf{u}_i)\in O_j} \begin{pmatrix} V_{\mathbf{c}}(\mathbf{x}_i) > 0 \;\wedge \\ \nabla V_{\mathbf{c}} \cdot f(\mathbf{x}_i, \mathbf{u}_i) < 0 \end{pmatrix} \end{array} \right\}. \tag{11}$$

The overall procedure is shown in Fig. 5. The procedure starts with an empty set $O_0 = \emptyset$ and the corresponding set of compatible function parameters $C_0 : \{\mathbf{c} \in C \mid V_{\mathbf{c}}(\mathbf{0}) = 0\}$. Each iteration j (starting from $j = 1$) involves the following steps:

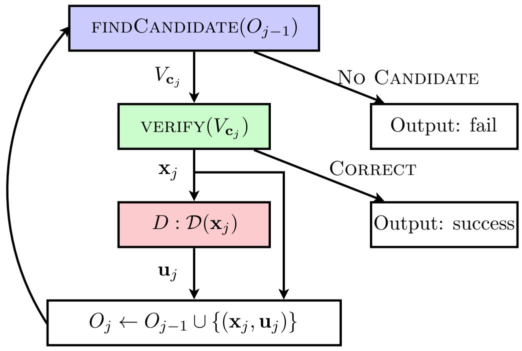


Fig. 5 Visualization of the learning framework

1. findCandidate: The learner checks if there exists a $V_{\mathbf{c}}$ compatible with O_{j-1}.
   (a) If no such $\mathbf{c}$ exists, the learner declares failure ($C_{j-1} = \emptyset$).
   (b) Otherwise, a candidate $\mathbf{c}_j \in C_{j-1}$ is chosen and the corresponding function $V_{\mathbf{c}_j}(\mathbf{x}) : \mathbf{c}_j.\mathbf{g}(\mathbf{x})$ is considered for verification.
2. verify: The verifier oracle tests whether $V_{\mathbf{c}_j}$ is a CLF (Eq. (9))
   (a) If yes, the process terminates successfully ($V_{\mathbf{c}_j}$ is a CLF)
   (b) Otherwise, the oracle provides a witness $\mathbf{x}_j \neq \mathbf{0}$ for the negation of Eq. (9).
3. update: Using the demonstrator $\mathbf{u}_j : \mathcal{D}(\mathbf{x}_j)$, a new observation $(\mathbf{x}_j, \mathbf{u}_j)$ is added to the training set:

$$O_j : O_{j-1} \cup \{(\mathbf{x}_j, \mathbf{u}_j)\} \tag{12}$$

$$C_j :\; C_{j-1} \cap \left\{ \mathbf{c} \mid \begin{array}{c} V_{\mathbf{c}}(\mathbf{x}_j) > 0 \;\wedge \\ \nabla V_{\mathbf{c}} \cdot f(\mathbf{x}_j, \mathbf{u}_j) < 0 \end{array} \right\}. \tag{13}$$

Theorem 1 *The learning framework as described above has the following property:*

1. $\mathbf{c}_j \notin C_j$. *I.e., the candidate found at the j^{th} step is eliminated from further consideration.*
2. *If the algorithm succeeds at iteration j, then the output function $V_{\mathbf{c}_j}$ is a valid CLF for stabilization.*
3. *The algorithm declares failure at iteration j if and only if no linear combination of the basis functions is a CLF compatible with the demonstrator.*

Proof 1) Suppose that $\mathbf{c}_j \in C_j$. Then, $\mathbf{c}_j$ satisfies the following conditions (Eq. (13)):

$$V_{\mathbf{c}_j}(\mathbf{x}_j) > 0 \;\wedge\; \nabla V_{\mathbf{c}_j} \cdot f(\mathbf{x}_j, \mathbf{u}_j) < 0\,.$$

However, the verifier guarantees that $\mathbf{c}_j$ is a counterexample for Eq. (6). I.e.,

$$V_{\mathbf{c}_j}(\mathbf{x}_j) \leq 0 \;\vee\; \nabla V_{\mathbf{c}_j} \cdot f(\mathbf{x}_j, \mathbf{u}_j) \geq 0\,,$$

which is a contradiction. Therefore, $\mathbf{c}_j \notin C_j$.

2) The algorithm declares success if the verifier could not find a counterexample. In other words, $V_{\mathbf{c}_j}$ satisfies conditions of Eq. (6) and therefore a CLF.

3) The algorithm declares failure if $C_j = \emptyset$. On the other hand, by definition, C_j yields the set of all $\mathbf{c}$ s.t. $V_{\mathbf{c}}$ (which is linear combination of basis functions) is compatible with the observations O_j. Therefore, $C_j = \emptyset$ implies that that no linear combination of the basis functions is compatible with the O_j and therefore compatible with the demonstrator.

One possible choice of basis functions involves monomials $g_j(\mathbf{x}) :\ \mathbf{x}^{\alpha_j}$ wherein $|\alpha_j|_1 \leq D_V$ for some degree bound D_V for the learning concept (CLF). Inverse results suggest polynomial basis for Lyapunov functions are expressive enough for verification of exponentially stable, smooth nonlinear systems over a bounded region [68]. This, justifies using polynomial basis for CLF.

In the next two section we present implementations of each of the modules involved, namely the learner and the verifier.

4 Learner

Recall that the learner needs to check if there exists a $\mathbf{c}$ s.t. $V_{\mathbf{c}}$ is compatible with the observation set O (Definition 5). In other words, we wish to check

$$(\exists \mathbf{c} \in \mathcal{C})\ V_{\mathbf{c}}(\mathbf{0}) = 0 \wedge \bigwedge_{(\mathbf{x}_i, \mathbf{u}_i) \in O} \left(\begin{array}{c} V_{\mathbf{c}}(\mathbf{x}_i) > 0 \wedge \\ \nabla V_{\mathbf{c}} \cdot f(\mathbf{x}_i, \mathbf{u}_i) < 0 \end{array} \right).$$

Note that each function $V_{\mathbf{c}}(\mathbf{x}_i) : \mathbf{c}^t \cdot \mathbf{g}(\mathbf{x}_i)$ in our hypothesis space, is linear in $\mathbf{c}$. Also, $\nabla V_{\mathbf{c}}.f(\mathbf{x}_i, \mathbf{u}_i)$ is linear in $\mathbf{c}$:

$$\nabla V_{\mathbf{c}}.f(\mathbf{x}_i, \mathbf{u}_i) = \sum_{k=1}^{r} c_k \nabla g_k(\mathbf{x}_i).f(\mathbf{x}_i, \mathbf{u}_i).$$

The (initial) space of all candidates C is assumed to be a hyper-rectangular box, and therefore a polytope. Let $\overline{C_j}$ represent the topological closure of the set C_j obtained at the j^{th} iteration (see Eq. (11)).

Lemma 1 *For each $j \geq 0$, $\overline{C_j}$ is a polytope.*

Proof We prove by induction. Initially C is an hyper-rectangular box. Also, $C_0 : C \cap H_0$, where

$$H_0 = \{\mathbf{c} \mid V_{\mathbf{c}}(\mathbf{0}) = \sum_{k=1}^{r} c_k g_k(\mathbf{0}) = 0\}.$$

As $V_{\mathbf{c}}$ is linear in $\mathbf{c}$, $H_0 : \{\mathbf{c} \mid \mathbf{a}_0^t.\mathbf{c} = b_0\}$ is a hyper-plane, where $\mathbf{a}_0$ and b_0 depend on the values of, $g_k(\mathbf{0})$ ($k = 1, \ldots, r$). And C_0 would be intersection of a polytope and a hyper-plane, which is a polytope. Now, assume $\overline{C_{j-1}}$ is a polytope. Recall that C_j is defined as $C_j : C_{j-1} \cap H_j$ (Eq. (13)), where

$$H_j :\ \left\{ \mathbf{c} \mid \begin{array}{c} \sum_{k=1}^{r} (c_k\ g_k(\mathbf{x}_j)) > 0 \wedge \\ \sum_{k=1}^{r} (c_k\ \nabla g_k(\mathbf{x}_j) \cdot f(\mathbf{x}_j, \mathbf{u}_j)) < 0 \end{array} \right\}.$$

Notice that $f(\mathbf{x}_j, \mathbf{u}_j)$, $g_k(\mathbf{x}_i)$, and $\nabla g_k(\mathbf{x}_i)$ are constants and

$$\begin{aligned} H_j &: H_{j1} \cap H_{j2} \\ H_{j1} &: \{\mathbf{c} \mid \mathbf{a}_{j1}^t.\mathbf{c} > b_{j1}\} \\ &= \{\mathbf{c} | \sum_{k=1}^{r} (c_k\ g_k(\mathbf{x}_j)) > 0\} \\ H_{j2} &: \{\mathbf{c} \mid \mathbf{a}_{j2}^t.\mathbf{c} > b_{j2}\} \\ &= \{\mathbf{c} | \sum_{k=1}^{r} (c_k\ \nabla g_k(\mathbf{x}_j) \cdot f(\mathbf{x}_j, \mathbf{u}_j)) < 0\}. \end{aligned}$$

Therefore, $\overline{C_j}$ is intersection of a polytope ($\overline{C_{j-1}}$) and two half-spaces (H_j) which yields another polytope.

The learner should sample a point $\mathbf{c}_j \in C_{j-1}$ at j^{th} iteration, which is equivalent to checking emptiness of a polytope with some strict inequalities. This is solved using slight modification of simplex method, using infinitesimals for strict inequalities, or using interior point methods [100]. We will now demonstrate that by choosing $\mathbf{c}_j$ carefully, we can guarantee the polynomial time termination of our learning framework.

4.1 Termination

Recall that in the framework, the learner provides a candidate and the verifier refutes the candidate by a counterexample and a new observation is generated by the demonstrator. The following lemma relates the sample $\mathbf{c}_j \in C_{j-1}$ at the j^{th} iteration and the set C_j in the subsequent iteration.

Lemma 2 *There exists a half-space $H_j^* :\ \mathbf{a}^t\mathbf{c} \geq b$ such that (a) $\mathbf{c}_j$ lies on boundary of hyperplane H_j^*, and (b) $C_j \subseteq C_{j-1} \cap H_j^*$.*

Proof Recall that we have $\mathbf{c}_j \in C_{j-1}$ but $\mathbf{c}_j \notin C_j$ by Theorem 1. Let $\hat{H}_j :\ \mathbf{a}^t\mathbf{c} = \hat{b}$ be a separating hyperplane between the (convex) set C_j and the point $\mathbf{c}_j$, such that $C_j \subseteq \{\mathbf{c} \mid \mathbf{a}^t\mathbf{c} \geq \hat{b}\}$. By setting the offset $b :\ \ \mathbf{a}^t\mathbf{c}_j$, we note that $b \leq \hat{b}$. Therefore, by defining H_j^* as $\mathbf{a}^t\mathbf{c} \geq b$, we obtain the required half-space that satisfies conditions (a) and (b).

While sampling a point from C_{j-1} is solved by solving a linear programming problem, Lemma. 2 suggests that the choice of $\mathbf{c}_j$ governs the convergence of the algorithm. Figure. 6 demonstrates the importance of this choice by showing candidate $\mathbf{c}_j$, hyperplanes H_{j1} and H_{j2} and C_j.

For a faster termination, we wish to remove a "large portion" of C_{j-1} to obtain a "smaller" C_j. There are two important factors which affect this: (i) counterexample $\mathbf{x}_j$ selection and (ii) candidate $\mathbf{c}_j$ selection. Counterexample $\mathbf{x}_j$, would affect $\mathbf{u}_j : \mathcal{D}(\mathbf{x}_j)$, $g(\mathbf{x}_j)$, and $f(\mathbf{x}_j, \mathbf{u}_j)$ and therefore defines the hyper-planes H_{j1} and H_{j2}. On the other hand, candidate $\mathbf{c}_j \notin C_j$. We postpone discussion on the counterexample selection to the next section, and for the rest of this section we focus on different techniques to generate a candidate $\mathbf{c}_j \in C_{j-1}$.

The goal is to find a $\mathbf{c}_j$ s.t.

$$\mathrm{Vol}(C_j) \leq \alpha \mathrm{Vol}(C_{j-1}) \,, \tag{14}$$

for each iteration j and a fixed constant $0 \leq \alpha < 1$, independent of the hyperplanes H_{j1} and H_{j2}. Here $\mathrm{Vol}(C_j)$ represents the volume of the (closure) of the set C_j. Since closure of C_j is contained in C which happens to be compact, this volume will always be finite. Note that if we can guarantee Eq. (14), it immediately follows that $\mathrm{Vol}(C_j) \leq \alpha^j \mathrm{Vol}(C_0)$. This implies that the volume of the remaining candidates "vanishes" rapidly.

Remark 4 By referring to $\mathrm{Vol}(C_j)$, we are implicitly assuming that C_j is not embedded inside a subspace of $\mathbb{R}^r$, i.e., it is full-dimensional. However, this assumption is not strictly true. Specifically, $C_0 : C \cap H_0$, where H_0 is a hyper-plane. Thus, strictly speaking, the volume of C_0 in $\mathbb{R}^r$ is 0. This issue is easily addressed by first factoring out the linearity space of C_0, i.e., the affine hull of C_0. This is performed by using the equality constraints that describe the affine hull to eliminate variables from C_0. Subsequently, C_0 can be treated as a full dimensional polytope in $\mathbb{R}^{r-d_j}$, wherein d_j is the dimension of its linearity space.

Furthermore, since $C_j \subseteq C_0$, we can continue to express C_j inside $\mathbb{R}^{r-d_j}$ using the same basis vectors as C_0. A further complication arises if C_j is embedded inside a smaller subspace. We do not treat this case in our analysis. However, note that this can happen for at most r iterations and thus, does not pose a problem for the termination analysis.

Intuitively, it is clear from Figure 6 that a candidate at the *center* of C_{j-1} would be a good one. We now relate the choice of $\mathbf{c}_j$ to an appropriate definition of center, so that Eq. (14) is satisfied.

Center of Maximum Volume Ellipsoid Maximum volume ellipsoid (MVE) inscribed inside a polytope is unique with many useful characteristics.

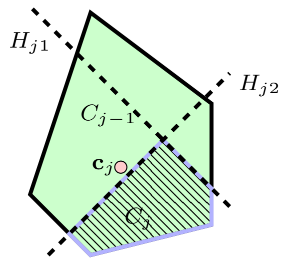


Fig. 6 Search space: Original candidate region C_j (green) at the start of the j^{th} iteration, the candidate $\mathbf{c}_j$, and the new region C_{j+1} (hatched region with blue lines).

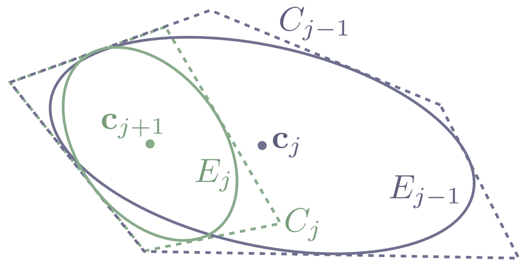


Fig. 7 Search Space: Original candidate region C_{j-1} (C_j) is shown in blue (green) polygon. The maximum volume ellipsoid E_{j-1} (E_j) is inscribed in C_{j-1} (C_j) and its center is the candidate $\mathbf{c}_j$ ($\mathbf{c}_{j+1}$).

Theorem 2 (Tarasov et al.[95]) *Let $\mathbf{c}_j$ be chosen as the center of the MVE inscribed in C_{j-1}. Then,*

$$Vol(C_j) \leq \left(1 - \frac{1}{r}\right) Vol(C_{j-1}) \,.$$

Recall, here that r is the number of basis functions such that $C_{j-1} \subseteq \mathbb{R}^r$. This leads us to a scheme that guarantees termination of the overall procedure in finitely many steps under some assumptions. The idea is simple. Select the center of the MVE inscribed in C_{j-1} at each iteration (Fig. 7).

Let $C \subseteq (-\Delta, \Delta)^r$ for $\Delta > 0$. Furthermore, let us additionally terminate the procedure as having failed whenever the $\mathrm{Vol}(C_j) < (2\delta)^r$ for some arbitrarily small $\delta > 0$. This additional termination condition is easily justified when one considers the precision limits of floating point numbers and sets of small volumes. Clearly, as the volume of the sets C_j decreases exponentially, each point inside the set will be quite close to one that is outside, requiring high precision arithmetic to represent and sample from the sets C_j.

Theorem 3 *If at each step $\mathbf{c}_j$ is chosen as the center of the MVE in C_{j-1}, the learning loop terminates in at most*

$$\frac{r(\log(\Delta) - \log(\delta))}{-\log\left(1 - \frac{1}{r}\right)} = O(r^2) \text{ iterations}.$$

Proof Initially, $\mathrm{Vol}(C_0) < (2\Delta)^r$. Then by Theorem 2

$$\mathrm{Vol}(C_j) \leq (1-\frac{1}{r})^j\ \mathrm{Vol}(C_0) < (1-\frac{1}{r})^j(2\Delta)^r$$
$$\implies \log\left(\frac{\mathrm{Vol}(C_j)}{(2\Delta)^r}\right) < j\ \log(1-\frac{1}{r})\,.$$

After $k = \frac{r(\log(\Delta)-\log(\delta))}{-\log(1-\frac{1}{r})}$ iterations:

$$\log\left(\frac{\mathrm{Vol}(C_j)}{(2\Delta)^r}\right) < \frac{r(\log(\Delta)-\log(\delta))}{-\log(1-\frac{1}{r})}\ \log(1-\frac{1}{r})\,,$$

and

$$\implies \log\left(\frac{\mathrm{Vol}(C_j)}{(2\Delta)^r}\right) < r\log\left(\frac{\delta}{\Delta}\right)$$
$$\implies \log\left(\frac{\mathrm{Vol}(C_j)}{(2\Delta)^r}\right) < r\log\left(\frac{2\delta}{2\Delta}\right)$$
$$\implies \log(\mathrm{Vol}(C_k)) < \log((2\delta)^r)\,.$$

And it is concluded that $\mathrm{Vol}(C_k) < (2\delta)^r$, which is the termination condition. And asymptotically $-\log(1-\frac{1}{r})$ is $\Omega(\frac{1}{r})$ (can be shown using Taylor expansion as $r \to \infty$) and therefore, the maximum number of iterations would be $O(r^2)$.

However, checking the termination condition is computationally expensive as calculating the volume of a polytope is $\sharp P$ hard, i.e., as hard as counting the number of solutions to a SAT problem. One solution is to first calculate an upper bound on the number of iterations using Theorem 3, and stop if the number of iterations has exceeded the upper-bound.

A better approach is to consider some robustness for the candidate.

Definition 7 (Robust Compatibility) A candidate $\mathbf{c}$ is δ-robust for $\delta > 0$ w.r.t. observations (demonstrator), iff for each $\hat{\mathbf{c}} \in \mathcal{B}_\delta(\mathbf{c})$, $V_{\hat{\mathbf{c}}} : \hat{\mathbf{c}}^t \cdot \mathbf{g}(\mathbf{x})$ is compatible with observations (demonstrator) as well.

Let E_j be the MVE inscribed inside C_j (Fig. 7). Following the robustness assumption, it is sufficient to terminate the procedure whenever:

$$\mathrm{Vol}(E_j) < \gamma\delta^r\,, \tag{15}$$

where γ is the volume of r-ball with radius 1.

Theorem 4 ([95, 42]) *Let $\mathbf{c}_j$ be chosen as the center of E_{j-1}. Then,*

$$Vol(E_j) \leq \left(\frac{8}{9}\right) Vol(E_{j-1})\ .$$

Theorem 5 *If at each step $\mathbf{c}_j$ is chosen as the center of E_{j-1}, the learning loop condition defined by Eq. (15) is violated in at most*

$$\frac{r(\log(\Delta)-\log(\delta))}{-\log\left(\frac{8}{9}\right)} = O(r)\ \textit{iterations}.$$

Proof Initially, $\mathcal{B}_\Delta(\mathbf{0})$ is the MVE inside box $[-\Delta, \Delta]^r$ and therefore, $\mathrm{Vol}(E_0) < \gamma\Delta^r$. Then by Theorem 2

$$\mathrm{Vol}(E_j) \leq (\frac{8}{9})^j\ \mathrm{Vol}(E_0) < (\frac{8}{9})^j\gamma\Delta^r$$
$$\implies \log(\mathrm{Vol}(E_j)) - \log(\gamma\Delta^r) < j\ \log(\frac{8}{9})\,.$$

After $k = \frac{r(\log(\Delta)-\log(\delta))}{-\log(\frac{8}{9})}$ iterations:

$$\log(\mathrm{Vol}(E_k)) - \log(\gamma\Delta^r) < \frac{r(\log(\Delta)-\log(\delta))}{-\log(\frac{8}{9})}\ \log(\frac{8}{9})\,,$$

and

$$\implies \log(\mathrm{Vol}(E_k)) - \log(\gamma\Delta^r) < r(\log(\delta)-\log(\Delta))$$
$$\implies \log(\mathrm{Vol}(E_k)) - \log(\gamma\Delta^r) < \log(\gamma\delta^r) - \log(\gamma\Delta^r)$$
$$\implies \log(\mathrm{Vol}(E_k)) < \log(\gamma\delta^r)\,.$$

It is concluded that $\mathrm{Vol}(E_k) < \gamma\delta^r$, which is the termination condition. And asymptotically the maximum number of iterations would be $O(r)$.

Volume of an ellipsoid is effectively computable and thus, such termination condition can be checked easily. Also, the convergence rate is linear in r as opposed to r^2, when the robustness is not guaranteed.

Theorem 6 *The learning framework either finds a control Lyapunov functions or proves that no linear combination of the basis function would yield a function with robust compatibility with the demonstrator.*

Proof By Theorem 1, if verifier certifies correctness of a solution V, then V is a CLF. Assume that the framework terminates after k iterations and no solution is found. Then, by Theorem 3, $\mathrm{Vol}(E_k) < \gamma\delta^r$. This means that a ball with radius δ would not fit in C_k as E_k is the MVE inscribed inside C_k. In other words

$$(\forall \mathbf{c} \in C_k)\ (\exists \hat{\mathbf{c}} \in \mathcal{B}_\delta(\mathbf{c}))\ \hat{\mathbf{c}} \notin C_k\,.$$

On the other hand, for all $\mathbf{c} \notin C_k$, $V_\mathbf{c}$ is not compatible with the observations O_j. Therefore, even if there is a CLF $V_\mathbf{c}$ s.t. $\mathbf{c} \in C_k$, the CLF is not robust in its compatibility with the demonstrator.

The MVE itself can be computed by solving a convex optimization problem[95, 99].

Other Definitions for Center of Polytope: Beside the center of MVE inscribed inside a polytope, there are other notions for defining center of a polytope. These include the center of gravity and Chebyshev center. Center of gravity provides the following inequality [14]

$$\mathrm{Vol}\,(C_j) \leq \left(1-\frac{1}{e}\right)\mathrm{Vol}\,(C_j) < 0.64\ \mathrm{Vol}(C_{j-1})\,,$$

meaning that the volume of candidate set is reduced by at least 36% at each iteration. Unfortunately, calculating center of gravity is very expensive. Chebyshev center [26] of a polytope is the center of the largest Euclidean ball that lies inside the polytope. Finding a Chebyshev center for a polytope is equivalent to solving a linear program, and while it yields a good heuristic, it would not provide an inequality in the form of Eq. (14).

There are also notions for defining center for a set of constraints, including analytic center, and volumetric center. Assuming $C : \{\mathbf{c} \mid \bigwedge_i \mathbf{a}_i^t.\mathbf{c} < b_i\}$, then analytic center for $\bigwedge_i \mathbf{a}_i^t.\mathbf{c} < b_i$ is defined as

$$ac(\bigwedge_i \mathbf{a}_i^t.\mathbf{c} < b_i) = \operatorname*{argmin}_{\mathbf{c}} - \sum_i \log(b_i - \mathbf{a}_i^t.\mathbf{c})\,.$$

Notice that infinitely many inequalities can represent C and any point inside C can be an analytic center depending on the inequalities. Atkinson et al. [8] and Vaidya [98] provide candidate generation techniques, based on these centers , along with appropriate termination conditions and convergence analysis.

5 Verifier

The verifier checks the CLF conditions in Eq. (9) for a candidate $V_{\mathbf{c}_j}(\mathbf{x}) : \mathbf{c}_j^t \cdot \mathbf{g}(\mathbf{x})$. Since the CLF is generated by the learner, it is guaranteed that $V_{\mathbf{c}_j}(\mathbf{0}) = 0$ (Eq. (11)). Accordingly, verification is split into two separate checks:

(A) Check if $V_{\mathbf{c}_j}(\mathbf{x})$ is a positive polynomial for $\mathbf{x} \neq \mathbf{0}$, or equivalently:

$$(\exists\ \mathbf{x} \neq \mathbf{0})\ V_{\mathbf{c}_j}(\mathbf{x}) \leq 0\,. \tag{16}$$

(B) Check if the Lie derivative of $V_{\mathbf{c}_j}$ can be made negative for each $\mathbf{x} \neq \mathbf{0}$ by a choice $\mathbf{u} \in U$:

$$(\exists \mathbf{x} \neq \mathbf{0})\ (\forall \mathbf{u} \in U)\ (\nabla V_{\mathbf{c}_j}) \cdot f(\mathbf{x}, \mathbf{u}) \geq 0\,. \tag{17}$$

This problem *seems* harder due to the presence of a *quantifier alternation.*

Lemma 3 *Eq.* (17) *holds for some* $\mathbf{x} \neq \mathbf{0}$ *iff*

$$\begin{aligned}(\exists\ \mathbf{x} \neq \mathbf{0}, \lambda)\ \lambda \geq \mathbf{0}, \lambda^t\mathbf{b} \geq -\nabla V_{\mathbf{c}_j}.f_0(\mathbf{x}) \\ A_i^t\lambda = \nabla V_{\mathbf{c}_j}.f_i(\mathbf{x})(i \in \{1 \ldots m\}).\end{aligned} \tag{18}$$

Proof Suppose Eq. (17) holds. Then, for the given V, there exists a $\mathbf{x} \neq \mathbf{0}$ s.t.

$$(\forall \mathbf{u} \in U)\ \nabla V \cdot f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \nabla V \cdot f_0(\mathbf{x}) + \\ \sum_{i=1}^{m} \nabla V \cdot f_i(\mathbf{x}) u_i \end{pmatrix} \geq 0, \tag{19}$$

which is equivalent to:

$$(\nexists \mathbf{u}) A\mathbf{u} \geq \mathbf{b} \wedge \nabla V \cdot f_0(\mathbf{x}) + \sum_{i=1}^{m} \nabla V \cdot f_i(\mathbf{x}) u_i < 0\,.$$

This yields a set of linear inequalities (w.r.t. $\mathbf{u}$). Using Farkas lemma, this is equivalent to

$$\begin{aligned}&(\exists \lambda \geq 0) A_i^t \lambda = \nabla V \cdot f_i(\mathbf{x})(i \in \{1...m\}) \\ &\qquad \lambda^t \mathbf{b} \geq -\nabla V \cdot f_0(\mathbf{x}).\end{aligned}$$

Thus, for a given V, Eq. (17) is equivalent to Eq. (18).

The verifier needs to check Eq. (16) and Eq. (18). This problem is *in general* undecidable if the basis functions include trigonometric and exponential functions. However, δ-decision procedures can solve these problems approximately [31]. In our experience, δ-decision procedures do not scale as verifiers for the range of benchmarks we wish to tackle. Nevertheless, these solvers allow us to conveniently implement a verifier for small but hard problems involving rational and trigonometric functions.

Assuming that the dynamics and chosen bases are polynomials in $\mathbf{x}$, the verification problem reduces to checking if a given semi-algebraic set defined by polynomial inequalities is empty. The verification problem for polynomial dynamics and polynomial CLFs is decidable with a high complexity (NP hard) [9]. Exact approaches using semi-algebraic geometry [17] or branch-and-bound solvers (including the dReal approach cited above) can tackle this problem precisely. However, for scalability, we consent to a relaxation using SDP solvers. We now present a relaxation using semidefinite programming (SDP) solvers.

5.1 SDP Relaxation

Let $\mathbf{w} :\ [\mathbf{x}, \lambda]$ collect the state variables $\mathbf{x}$ and the dual variables λ involved in the conditions stated in (18). The core idea behind the SDP relaxation is to consider a vector collecting all monomials of degree up to D:

$$\mathbf{m} : \begin{pmatrix} 1 \\ w_1 \\ w_2 \\ \ldots \\ \mathbf{w}^D \end{pmatrix},$$

wherein D is chosen to be at least half of the maximum degree in $\mathbf{x}$ among all monomials in $g_j(\mathbf{x})$ and $\nabla g_j \cdot f_i(\mathbf{x})$:

$$D \geq \frac{1}{2} \max \left(\bigcup_j \left(\{\deg(g_j)\} \cup \{\bigcup_i \deg(\nabla g_j \cdot f_i)\} \right) \right).$$

Let us define $Z(\mathbf{w}) : \mathbf{m}\mathbf{m}^t$, which is a symmetric matrix of monomial terms of degree at most $2D$. Each polynomial of degree up to $2D$ may now be written as a trace inner product

$$p(\mathbf{x}, \lambda) : \langle P, Z(\mathbf{w}) \rangle = \mathsf{trace}(P Z(\mathbf{w}))\,,$$

wherein the matrix P has real-valued entries that define the coefficients in p corresponding to the various monomials. Although, Z is a function of $\mathbf{x}$ and λ, we will write $Z(\mathbf{x})$ as a function of just $\mathbf{x}$ to denote the matrix $Z([\mathbf{x}, \mathbf{0}])$ (i.e., set $\lambda = \mathbf{0}$).

Checking Eq. (16) is equivalent to solving the following optimization problem over $\mathbf{x}$

$$\begin{array}{ll} \mathsf{max}_{\mathbf{x}} \, \langle I, Z(\mathbf{x}) \rangle & \\ \mathsf{s.t.} & \left\langle \mathcal{V}_{\mathbf{c}_j}, Z(\mathbf{x}) \right\rangle \leq 0\,, \end{array} \tag{20}$$

wherein I is the identity matrix, and $V_{\mathbf{c}_j}(\mathbf{x})$ is written in the inner product form as $\left\langle \mathcal{V}_{\mathbf{c}_j}, Z(\mathbf{x}) \right\rangle$. Let $\langle \Lambda_k, Z(\mathbf{w}) \rangle$ represent the variable λ_k. λ is represented as vector $\Lambda(Z(\mathbf{w}))$, wherein the k^{th} element is $\langle \Lambda_k, Z(\mathbf{w}) \rangle$. Then, the conditions in (18) are now written as

$$\begin{array}{ll} \mathsf{max}_{\mathbf{w}} \, \langle I, Z(\mathbf{w}) \rangle & \\ \mathsf{s.t.} & \left\langle F_{\mathbf{c}_j,i}, Z(\mathbf{w}) \right\rangle = A_i^t \Lambda(Z(\mathbf{w})),\ i \in \{1, \ldots, m\} \\ & \left\langle -F_{\mathbf{c}_j,0}, Z(\mathbf{w}) \right\rangle \leq \mathbf{b}^t \Lambda(Z(\mathbf{w})) \\ & \Lambda(Z(\mathbf{w})) \geq 0\,, \end{array} \tag{21}$$

wherein the components $\nabla V_{\mathbf{c}_j} \cdot f_i(\mathbf{x})$ defining the Lie derivatives of $V_{\mathbf{c}_j}$ are now written in terms of $Z(\mathbf{w})$ as $\left\langle F_{\mathbf{c}_j,i}, Z(\mathbf{w}) \right\rangle$. Notice that $Z(\mathbf{0})$ is a square matrix where the first element $(Z(\mathbf{0})_{1,1})$ is 1 and the rest of the entries are zero. Let $Z_0 = Z(\mathbf{0})$. Then $\langle I, Z_0 \rangle = 1$, and $(\forall \mathbf{w})\ Z(\mathbf{w}) \succeq Z_0$.

The SDP relaxation is used to solve these problems and provide an upper bound of the solution and D defines the degree of relaxation [34]. The relaxation treats $Z(\mathbf{w})$ as a fresh matrix variable Z that is no longer a function of $\mathbf{w}$. The constraint $Z \succeq Z_0$ is added. $Z(\mathbf{w}) : \mathbf{m}\mathbf{m}^t$ is a rank one matrix and ideally, Z should be constrained to be rank one as well. However, such a constraint is non-convex, and therefore, will be dropped from our relaxation. Also, constraints involving $Z(\mathbf{w})$ in Eqs. (20) and (21) are added as support constraints (cf. [47,48,34]). Both optimization problems (Eqs.(20) and (21)) are feasible by setting Z to be Z_0. Furthermore, if the optimal solution for each problem is 1 in the SDP relaxation, then we will conclude that the given candidate is a CLF. Unfortunately, the converse is not necessarily true: the relaxation may fail to recognize that a given candidate is in fact a CLF.

Lemma 4 *Whenever the relaxed optimization problems in Eqs.* (20) *and* (21) *yield* 1 *as a solution, then the given candidate* $V_{\mathbf{c}_j}(\mathbf{x})$ *is in fact a CLF.*

Proof Suppose that $V_{\mathbf{c}_j}$ is not a CLF but both optimization problems yield an optimal value of 1. Then, one of Eq. (16) or Eq. (17) is satisfied. I.e. $(\exists \mathbf{x}^* \neq \mathbf{0}, \lambda^* \geq \mathbf{0})$ s.t. $V_{\mathbf{c}_j}(\mathbf{x}^*) \leq 0$ or $A_i^t \lambda^* = \nabla V_{\mathbf{c}_j} . f_i(\mathbf{x}^*) (i \in \{1 \ldots m\}), \lambda^{*t} \mathbf{b} \geq -\nabla V_{\mathbf{c}_j} . f_0(\mathbf{x}^*)$. Let $\mathbf{w}^* = [\mathbf{x}^*, \lambda^*]$ and therefore $Z(\mathbf{w}^*) \succeq Z_0$ is a solution for Eq. (20) or Eq. (21). Let $Z' = Z(\mathbf{x}^*) - Z_0$. As $\mathbf{w}^* \neq \mathbf{0}$, Z' has a non-zero diagonal element, and since $Z' \succeq 0$, we may also conclude that one of the eigenvalues of Z' must be positive. Therefore, $\langle I, Z' \rangle > 0$ as the trace of Z' is the sum of eigenvalues of Z'. Thus, $\langle I, Z(\mathbf{w}) \rangle > \langle I, Z_0 \rangle = 1$. Thus, the optimal solution of at least one of the two problems has to be greater than one. This contradicts our original assumption.

However, the converse is not true. It is possible for $Z \succeq Z_0$ to be optimal for either relaxed condition, but $Z \neq Z(\mathbf{w})$ for any $\mathbf{w}$. This happens because (as mentioned earlier) the relaxation drops two key constraints to convexify the conditions: (1) Z has to be a rank one matrix written as $Z : \mathbf{m}\mathbf{m}^t$ and (2) there is a $\mathbf{w}$ such that $\mathbf{m}$ is the vector of monomials corresponding to $\mathbf{w}$.

Lemma 5 *Suppose Eq.* (21) *has a solution* $Z \neq Z_0$*, then*

$$(\forall \mathbf{u} \in U)\ \left\langle F_{\mathbf{c}_j,0}, Z \right\rangle + \sum_{i=1}^{m} \left\langle F_{\mathbf{c}_j,i}, Z \right\rangle u_i \geq 0\,.$$

Proof While in the relaxed problem, the relation between monomials are lost, each inequality in Eq. (21) holds. Let $\hat{\lambda} = \Lambda(Z)$. Then, we have:

$$\begin{aligned} &\left\langle F_{\mathbf{c}_j,i}, Z \right\rangle = A_i^t \hat{\lambda},\ i \in \{1, \ldots, m\} \\ &\qquad \left\langle -F_{\mathbf{c}_j,0}, Z \right\rangle \leq \mathbf{b}^t \hat{\lambda},\ \hat{\lambda} \geq 0\,. \end{aligned}$$

Similar to Lemma. 3 (using Farkas Lemma) this is equivalent to

$$(\forall \mathbf{u} \in U)\ \left\langle F_{\mathbf{c}_j,0}, Z \right\rangle + \sum_{i=1}^{m} \left\langle F_{\mathbf{c}_j,i}, Z \right\rangle u_i \geq 0\,.$$

5.2 Lifting the Counterexamples

Thus far, we have observed that the relaxed optimization problems (Eqs. (20) and (21)) yield matrices Z as counterexamples, rather than vectors $\mathbf{x}$. Furthermore, given a solution Z, there is no way for us to extract a corresponding $\mathbf{x}$ for reasons mentioned above. We solve this issue by "lifting" our entire learning loop to work with observations of the form:

$$O_j : \{(Z_1, \mathbf{u}_1), \ldots, (Z_j, \mathbf{u}_j)\}\,,$$

effectively replacing states $\mathbf{x}_i$ by matrices Z_i.

Also, each basis function $g_k(\mathbf{x})$ in $\mathbf{g}$ is now written instead as $\langle G_k, Z\rangle$. The candidates are therefore, $\sum_{k=1}^{r} c_k \langle G_k, Z\rangle$. Likewise, we write the components of its Lie derivative $\nabla g_k \cdot f_i$ in terms of Z ($\langle G_{ki}, Z\rangle$). Therefore

$$\mathcal{V}_{\mathbf{c}} = \sum_{k=1}^{r} c_k G_k \ , \ F_{\mathbf{c},i} = \sum_{k=1}^{r} c_k G_{ki}\,. \tag{22}$$

Definition 8 (Relaxed CLF) A polynomial function $V_{\mathbf{c}}(\mathbf{x}) = \sum_{k=1}^{r} c_k g_k(\mathbf{x})$, s.t. $\langle \mathcal{V}_{\mathbf{c}}, Z_0\rangle = 0$ is a D-relaxed CLF iff for all $Z \neq Z_0$:

$$\begin{aligned}&\langle \mathcal{V}_{\mathbf{c}}, Z\rangle > 0 \ \wedge \\ &(\exists \mathbf{u} \in U)\ \ \langle F_{\mathbf{c},0}, Z\rangle + \textstyle\sum_{i=1}^{m} \langle F_{\mathbf{c},i}, Z\rangle < 0\,.\end{aligned} \tag{23}$$

Theorem 7 *A relaxed CLF is a CLF.*

Proof Suppose that $V_{\mathbf{c}}$ is not a CLF. The proof is complete by showing that $V_{\mathbf{c}}$ is not a relaxed CLF. If $V_{\mathbf{c}}(\mathbf{0}) \neq 0$, then $\langle \mathcal{V}_{\mathbf{c}}, Z_0\rangle \neq 0$ and $V_{\mathbf{c}}$ is not a relaxed CLF. Otherwise, according to Eq. (6) there exists a $\mathbf{x} \neq \mathbf{0}$ s.t.

$$V_{\mathbf{c}}(\mathbf{x}) \leq 0 \ \vee \ (\forall \mathbf{u} \in U)\ \nabla V_{\mathbf{c}}.f(\mathbf{x}, \mathbf{u}) \geq 0\,.$$

Therefore, there exists $\mathbf{x} \neq \mathbf{0}$ s.t.

$$\begin{aligned}&\langle \mathcal{V}_{\mathbf{c}}, Z(\mathbf{x})\rangle \leq 0 \ \vee \\ &(\forall \mathbf{u} \in U)\ \ \langle F_{\mathbf{c},0}, Z(\mathbf{x})\rangle + \sum_{i=1}^{m} \langle F_{\mathbf{c},i}, Z(\mathbf{x})\rangle\, u_i \geq 0\,.\end{aligned}$$

Setting $Z :\ Z(\mathbf{x})$ shows that $V_{\mathbf{c}}$ is not a relaxed CLF, since the negation of Eq. (23) holds.

We lift the overall formal learning framework to work with matrices Z as counterexamples using the following modifications to various parts of the framework:

1. First, for each $(Z_j, \mathbf{u}_j)$ in the observation set, Z_j is the feasible solution returned by the SDP solver while solving Eqs. (21) and (20).
2. However, the demonstrator $\mathcal{D}$ requires its input to be a state $\mathbf{x} \in X$. We define a projection operator $\pi : \zeta \mapsto X$ mapping each Z to a state $\mathbf{x} : \pi(Z)$, such that the demonstrator operates over $\pi(Z_j)$ at each step. Note that the vector of monomials $\mathbf{m}$ used to define Z from $\mathbf{x}$ includes the degree one terms $x_1, \ldots, x_n$. The projection operator simply selects the entries from Z corresponding to these variables. Other more sophisticated projections are also possible, but not considered in this work.
3. The space of all candidates C remains unaltered except that each basis polynomial is now interpreted as $g_j : \langle G_j, Z\rangle$ and similarly for the Lie derivative $(\nabla g_j) \cdot f(\mathbf{x}, \mathbf{u})$. Thus, the learner is effectively unaltered.

Definition 9 (Relaxed Observation Compatibility) A polynomial function $V_{\mathbf{c}}$ is said to be compatible with a set of D-relaxed-observations O iff $V_{\mathbf{c}}$ respects the D-relaxed CLF conditions (Eq. (6)) for every point in O:

$$\begin{aligned}&\langle \mathcal{V}_{\mathbf{c}}, Z_0\rangle = 0 \ \wedge \\ &\bigwedge_{(Z_k, \mathbf{u}_k) \in O_j} \left(\begin{array}{c} \langle \mathcal{V}_{\mathbf{c}}, Z_k\rangle > 0 \ \wedge \\ \langle F_{\mathbf{c},0}, Z_k\rangle + \sum_{i=1}^{m} \langle F_{\mathbf{c},i}, Z_k\rangle\, u_{ki} < 0 \end{array}\right)\,.\end{aligned}$$

Definition 10 (Relaxed Demonstrator Compatibility) A polynomial function $V_{\mathbf{c}}$ is said to be compatible with a relaxed-demonstrator $\mathcal{D} \circ \pi$ iff $V_{\mathbf{c}}$ respects the D-relaxed CLF conditions (Eq. (6)) for every observation that can be generated by the relaxed-demonstrator:

$$\begin{aligned}&\langle \mathcal{V}_{\mathbf{c}}, Z_0\rangle = 0 \ \wedge \\ &(\forall Z \succeq Z_0,\ Z \neq Z_0) \\ &\qquad\left(\begin{array}{c} \langle \mathcal{V}_{\mathbf{c}}, Z\rangle > 0 \ \wedge \\ \langle F_{\mathbf{c},0}, Z\rangle + \sum_{i=1}^{m} \langle F_{\mathbf{c},i}, Z\rangle\, \mathcal{D}(\pi(Z))_i < 0 \end{array}\right)\,.\end{aligned}$$

In other words, $V_{\mathbf{c}}$ is a relaxed Lyapunov function for the closed loop system $\Psi(X, U, f, \mathcal{D} \circ \pi)$.

Theorem 8 *The adapted formal learning framework terminates and either finds a CLF V, or proves that no linear combination of basis functions would yield a CLF, with robust compatibility w.r.t. the (relaxed) demonstrator.*

Proof C_{j-1} represents all $\mathbf{c}$ s.t. $V_{\mathbf{c}}$ is compatible with relaxed-observation O_{j-1}. Still $\mathcal{V}_{\mathbf{c}}$ and $F_{\mathbf{c},i}$ are linear in $\mathbf{c}$ (Eq. (22)), and therefore C_{j-1} which is the set of all $\mathbf{c} \in C$ s.t.

$$\begin{aligned}&\langle \mathcal{V}_{\mathbf{c}}, Z_0\rangle = 0 \ \wedge \\ &\bigwedge_{(Z_k, \mathbf{u}_k) \in O_{j-1}} \left(\begin{array}{c} \langle \mathcal{V}_{\mathbf{c}}, Z_k\rangle > 0 \ \wedge \\ \sum_{i=1}^{m} \langle F_{\mathbf{c},i}, Z_k\rangle\, u_{ki} + \langle F_{\mathbf{c},0}, Z_k\rangle < 0 \end{array}\right)\,,\end{aligned}$$

is a polytope (similar to Lemma 1). Suppose, at j^{th} iteration, $V_{\mathbf{c}_j} : \mathbf{c}_j^t.\mathbf{g}$ is generated by the learner. The relaxed verifier solves Eqs. (20) and (21). If the optimal solution for these problems are 1, by Lemma 4, $V_{\mathbf{c}_j}$ is a CLF. Otherwise, it returns a counterexample $Z_j \succeq Z_0$ and $Z_j \neq Z_0$. More over, according to Eqs. (20) and (21) and Lemma 5:

$$\begin{aligned}&\left\langle \mathcal{V}_{\mathbf{c}_j}, Z_j\right\rangle \leq 0 \ \vee \\ &(\forall \mathbf{u} \in U)\ \ \left\langle F_{\mathbf{c}_j,0}, Z_j\right\rangle + \sum_{i=1}^{m} \left\langle F_{\mathbf{c}_j,i}, Z_j\right\rangle u_i \geq 0\,.\end{aligned}$$

In other words, $V_{\mathbf{c}_j}$ is not a D-relaxed CLF. Next, the demonstrator generates a proper feedback for $\pi(Z_j)$ and observation $(Z_j, \mathcal{D}(\pi(Z_j)))$ is added to the set of observations. Notice that $V_{\mathbf{c}_j}$ does not respect the D-relaxed CLF conditions for $(Z_j, \mathcal{D}(\pi(Z_j)))$. I.e.

$$\langle \mathcal{V}_{\mathbf{c}_j}, Z_j \rangle \leq 0 \vee$$
$$\langle F_{\mathbf{c}_j,0}, Z_j \rangle + \sum_{i=1}^{m} \langle F_{\mathbf{c}_j,i}, Z_j \rangle \mathcal{D}(\pi(Z_j))_i \geq 0 \,.$$

Therefore, the new set C_j does not contain $\mathbf{c}_j$. Now, the learner uses the center of maximum volume ellipsoid, to generate the next candidate. This process repeats and the learning procedure terminates in finite iterations. When the algorithm returns with no solution, it means that $\mathrm{Vol}(C_j) \leq \gamma\delta^r$. Similar to Theorem 6, this guarantees that no ball of radius δ fits inside C_j, which represents the set of all linear combination of basis functions, compatible with the relaxed observations. Therefore, no linear combination of basis functions would yield a CLF with robust compatibility with the relaxed observation and therefore with the relaxed-demonstrator.

In the rest of this paper, we use CLF for discussions. Nevertheless, the same results can be applied to relaxed CLF as well.

5.3 Counterexamples Selection

As discussed earlier, in Section 4, there are two important factors that affect the overall convergence rate of the learning framework: (a) the choice of a candidate $\mathbf{c}_j \in C_{j-1}$ and (b) the choice of a counterexample $\mathbf{x}_j$ that shows that the current candidate $V_{\mathbf{c}_j}$ is not a CLF. We will now discuss the choice of a "good" counterexample.

As mentioned, when there is a counterexample $\mathbf{x}_j$ for $V_{\mathbf{c}_j}$, there are two half spaces $H_{j1} : \{\mathbf{c} \mid \mathbf{a}_{j1}^t.\mathbf{c} > b_{j1}\}$, and $H_{j2} : \{\mathbf{c} \mid \mathbf{a}_{j2}^t.\mathbf{c} > b_{j2}\}$ such that $C_j : C_{j-1} \cap H_{j1} \cap H_{j2}$. In particular, $\mathbf{c}_j \notin C_j$, yields the following constraints over $\mathbf{c}_j$:

$$\mathbf{a}_{j1}^t.\mathbf{c}_j \leq b_{j1} \vee \mathbf{a}_{j2}^t.\mathbf{c}_j \leq b_{j2} \,. \tag{24}$$

In general, the counterexample affects the coefficients of the half-spaces $\mathbf{a}_{jl}, b_{jl}$ for $l \in \{1, 2\}$. To wit, the counterexample $\mathbf{x}_j$ defines values for $\mathbf{u}_j : \mathcal{D}(\mathbf{x}_j)$, $g_i(\mathbf{x}_j)$, $f_i(\mathbf{x}_j, \mathbf{u}_j)$, which in turn, define H_{j1} and H_{j2}. Thus, a good counterexample should "remove" as large a set as possible from C_{j-1}. Looking at Eq. (24), it is clear that $\mathbf{a}_{jl}^t.\mathbf{c}_j - b_{jl}$ would measure how "far away" the counterexample is from the boundary of the half-space H_{jl}, assuming that $||\mathbf{a}_{jl}||$ is kept constant. As proposed in our earlier work [77], one could find a counterexample that maximizes these quantities, so that a "good" counterexample can be selected. For checking (16), the verifier finds a counterexample $\mathbf{x}$ that maximizes a slack variable γ s.t.

$$V_{\mathbf{c}_j}(\mathbf{x}) \leq -\gamma \,,$$

and for the second check (18), the slack variable γ is introduced and maximized as follows:

$$\lambda \geq \gamma \ \wedge \ \bigwedge_{i=1}^{m} A_i^t \lambda = \nabla V_{\mathbf{c}_j} \cdot f_i(\mathbf{x}) \ \wedge$$
$$\lambda^t.\mathbf{b} \geq -\nabla V_{\mathbf{c}_j} \cdot f_0(\mathbf{x}) + \gamma \,.$$

As such, we cannot prove improved bounds on the number of iterations to terminate using this approach. However, we do, in fact, see a significant decrease in the number of iterations by adding an objective function to the selection of the counterexample.

6 Specifications

In previous sections, the problem of finding a CLF was discussed. However, the concept can be extend to other Lyapunov-like arguments that are useful for specifications such as reach-while-stay, and safety. In this section, some of these specifications are addressed.

6.1 Local Lyapunov Function

Many nonlinear systems are only locally stabilizable, especially in presence of input saturation. Therefore, we wish to study stabilization inside a compact set S. Let $int(R)$ be the interior of set R. We consider a compact and connected set $S \subset X$ where the origin $\mathbf{0} \in int(S)$ is the state we seek to stabilize to. Furthermore, we restrict the set S to be a basic semi-algebraic set defined by a conjunction of polynomial inequalities:

$$S : \{\mathbf{x} \in \mathbb{R}^n \mid p_{S,1}(\mathbf{x}) \leq 0, \ldots, p_{S,k}(\mathbf{x}) \leq 0\} \,.$$

The stabilization problem can be reduced to the problem of finding a local CLF V which respect the following constraints

$$\begin{array}{rl} & V(\mathbf{0}) = 0 \\ (\forall \mathbf{x} \in S \setminus \{\mathbf{0}\}) & V(\mathbf{x}) > 0 \\ (\forall \mathbf{x} \in S \setminus \{\mathbf{0}\}) \ (\exists \mathbf{u} \in U) & \nabla V \cdot f(\mathbf{x}, \mathbf{u}) < 0 \,. \end{array} \tag{25}$$

Given a function V and a comparison predicate $\bowtie \in \{=, \leq, <, \geq, >\}$, we define $V^{\bowtie \beta}$ as the set:

$$V^{\bowtie \beta} = \{\mathbf{x} | V(\mathbf{x}) \bowtie \beta\} \,.$$

Let β^* be maximum β s.t. $V^{\leq\beta} \subseteq S$. Having a CLF V, it guarantees that there is a strategy to keep the state inside $V^{<\beta}$, and stabilize to the origin (Fig. 4).

Theorem 9 *Given a control affine system Ψ, where $U : \mathbb{R}^m$ and a polynomial control Lyapunov function V satisfying Eq. (25), there is a feedback function $\mathcal{K}$ for which if $\mathbf{x}_0 \in V^{<\beta^*}$, then:*

1. $(\forall t \geq 0)\ \mathbf{x}(t) \in S$
2. $(\forall \epsilon > 0)\ (\exists T \geq 0)\ \|\mathbf{x}(T) - \mathbf{0}\| < \epsilon$.

Proof First, using Sontag results, there exists a feedback function $\mathcal{K}^*$ s.t. while $\mathbf{x} \in S$, then $\frac{dV}{dt} = \nabla V \cdot f(\mathbf{x}, \mathbf{u}) < 0$ [89]. Assuming $\mathbf{x}(0) = \mathbf{x}_0 \in V^{<\beta^*} \subset S$, then initially $V(\mathbf{x}(0)) < \beta^*$. Now, assume the state reaches ∂S at time t_2. By continuity, there is a time $t_1 \leq t_2$ s.t. $\mathbf{x}(t_1) \in \partial(V^{<\beta^*})$ and $(\forall t \in [0, t_1])\ \mathbf{x}(t) \in S$. Thus, $V(\mathbf{x}(t_1)) = \beta^*$ and

$$V(\mathbf{x}(t_1)) = \left(V(\mathbf{x}(0)) + \int_0^{t_1} \frac{dV}{dt} dt \right) < V(\mathbf{x}(0)) .$$

This means $V(\mathbf{x}(t_1)) < \beta^*$, which is a contradiction. Therefore, the state never reaches ∂S and remains in $int(S)$ forever.

V would be a Lyapunov function for the closed loop system when the control unit is replaced with the feedback function $\mathcal{K}^*$ and using standard results in Lyapunov theory $(\forall \epsilon > 0)\ (\exists T \geq 0)\ ||\mathbf{x}(T) - 0|| < \epsilon$.

Finding a local CLF is similar to finding a global one. One only needs to consider set S in the formulation. The observation set would consists of $(\mathbf{x}_i, \mathbf{u}_i)_{i=1}^j$ where $\mathbf{x}_i$ is inside S and the verifier would check the following conditions:

$$(\exists \mathbf{x} \neq \mathbf{0}) \bigwedge_{i=1}^{k} p_{S,i}(\mathbf{x}) \leq 0 \wedge V(\mathbf{x}) \geq 0$$
$$(\exists \mathbf{x} \neq \mathbf{0}) \bigwedge_{i=1}^{k} p_{S,i}(\mathbf{x}) \leq 0 \wedge (\forall \mathbf{u} \in U)\ \nabla V \cdot f(\mathbf{x}, \mathbf{u}) \geq 0 ,$$

which is as hard as the one solved in Section. 5.

Lemma 6 *Assuming (i) the demonstrator function $\mathcal{D}$ is smooth, (ii) the closed loop system with feedback law $\mathcal{D}$ is exponentially stable over a bounded region S, then there exists a local polynomial CLF, compatible with $\mathcal{D}$.*

Proof Under assumption (i) and (ii), one can show that a polynomial local Lyapunov function V (not control Lyapunov function) exists for the closed loop system $\Psi(X, U, f, \mathcal{D})$ [68]:

$$V(\mathbf{0}) = 0\ \wedge (\forall \mathbf{x} \in S \setminus \mathbf{0}) \begin{pmatrix} V(\mathbf{x}) > 0 \\ \nabla V \cdot f(\mathbf{x}, \mathcal{D}(\mathbf{x})) < 0 \end{pmatrix} .$$

This means that V is compatible with the demonstrator. V is also a local CLF as it satisfies Eq. (25).

As mentioned, the learning framework fails when the basis functions are not expressive to capture a CLF compatible with the demonstrator and one needs to update the demonstrator and/or the set of basis functions. However, if one believes that the demonstrator satisfies the conditions in Lemma 6, then, success of the learning procedure is guaranteed, provided the set of basis functions is rich enough.

6.2 Barrier Certificate

Barrier certificates are used to guarantee safety properties for the system. More specifically, given compact and connected semi-algebraic sets S (safe) and I (initial) s.t. $I \subset int(S)$, the overall goal is to ensure that whenever $\mathbf{x}(0) \in I$, we have $\mathbf{x}(t) \in S$ for all time $t \geq 0$. The sets S, I are expressed as semi-algebraic sets of the following form:

$$S : \{\mathbf{x} \in \mathbb{R}^n \mid p_{S,1}(\mathbf{x}) \leq 0, \ldots, p_{S,k}(\mathbf{x}) \leq 0\}$$
$$I : \{\mathbf{x} \in \mathbb{R}^n \mid p_{I,1}(\mathbf{x}) \leq 0, \ldots, p_{I,l}(\mathbf{x}) \leq 0\} .$$

The safety problem can be reduced to the problem of finding a (relaxed [69]) control barrier certificate B which respect the following constraints [102]:

$$\begin{aligned} (\forall \mathbf{x} \in I)\ \ & B(\mathbf{x}) < 0 \\ (\forall \mathbf{x} \notin int(S))\ \ & B(\mathbf{x}) > 0 \\ (\forall \mathbf{x} \in S \setminus int(I))\ (\exists \mathbf{u} \in U)\ \ & \nabla B \cdot f(\mathbf{x}, \mathbf{u}) < 0 . \end{aligned} \tag{26}$$

To find such a barrier certificate, one needs to define B as a linear combination of basis functions and use the framework to find a correct B. The verifier would check the following conditions that negate each of the conditions in Eq. (26). First we check if there is a $\mathbf{x} \in I$ such that $B(\mathbf{x}) \geq 0$.

$$(\exists \mathbf{x}) \quad \bigwedge_{j=1}^{l} p_{I,j}(\mathbf{x}) \leq 0\ \ \wedge\ \ B(\mathbf{x}) \geq 0 .$$

Next, we check if there exists a $\mathbf{x} \notin int(S)$ such that $B(\mathbf{x}) \leq 0$. Clearly, if $\mathbf{x} \notin int(S)$, we have $p_{S,i}(\mathbf{x}) \geq 0$ for at least one $i \in \{1, \ldots, k\}$. This yields k conditions of the form:

$$(\exists \mathbf{x})\ p_{S,i}(\mathbf{x}) \geq 0 \wedge B(\mathbf{x}) \leq 0,\ \ i \in \{1, \ldots, k\} .$$

Finally, we ask if $\exists \mathbf{x} \in S \setminus int(I)$ that violates the decrease condition. Doing so, we obtain l conditions. For each $i \in \{1, \ldots, l\}$, we solve

$$(\exists \mathbf{x})\ \underbrace{p_{I,i}(\mathbf{x}) \geq 0}_{\mathbf{x} \notin int(I)} \wedge \underbrace{\bigwedge_{j=1}^{k} p_{S,j}(\mathbf{x}) \leq 0}_{\mathbf{x} \in S}$$
$$\wedge\ (\forall \mathbf{u} \in U)\ \nabla B \cdot f(\mathbf{x}, \mathbf{u}) \geq 0 ,$$

Overall, we have $1+k+l$ different checks. If any of these checks result in $\mathbf{x}$, it serves as a counterexample to the conditions for a barrier function (26).

As before, we choose basis functions $g_1,\ldots,g_r$ for the barrier set $B_{\mathbf{c}} : \sum_{k=1}^{r} c_k g_k(\mathbf{x})$. Given observations set $O_j : \{(\mathbf{x}_1,\mathbf{u}_1),\ldots,(\mathbf{x}_j,\mathbf{u}_j)\}$, the corresponding candidate set C_j of observation compatible barrier functions is defined as the following:

$$C_j : \left\{ \mathbf{c} \middle| \bigwedge_{(\mathbf{x}_i,\mathbf{u}_i)\in O_j} \begin{pmatrix} \mathbf{x}_i \in I \to B_{\mathbf{c}}(\mathbf{x}_i) < 0 \,\wedge \\ \mathbf{x}_i \not\in int(S) \to B_{\mathbf{c}}(\mathbf{x}_i) > 0 \,\wedge \\ \mathbf{x}_i \in S \setminus int(I) \\ \to \nabla B_{\mathbf{c}} \,.f(\mathbf{x}_i,\mathbf{u}_i) < 0 \end{pmatrix} \right\}.$$

The LHS of the implication for each observation $(\mathbf{x}_i,\mathbf{u}_i)$ is evaluated and the RHS constraint is added only when the LHS holds. Nevertheless, $\overline{C_j}$ remains a polytope similar to Lemma. 1.

Remark 5 For the original control barrier certificates, it is sufficient to check whether B can be decreased on the boundary ($B^{=0}$). The relaxed version of control barrier certificates is introduced by Prajna et al. [69] using sum of squares (SOS) relaxation. Here we use this relaxation to simplify the candidate generation process. However, for the verification process this relaxation is not needed and without any complication, one could verify the original conditions as opposed to the relaxed ones. This trick will improve the precision of the method.

6.3 Reach-While-Stay

In this problem, the goal is to reach a target set T from an initial set I, while staying in a safe set S, wherein $I \subseteq S$. The set S is assumed to be compact. By combining the local Lyapunov function and a barrier certificate, one can define a smooth, Lyapunov-like function V, that satisfies the following conditions (see [79]):

$$\begin{array}{ll} C1: & (\forall \mathbf{x} \in I)\ V(\mathbf{x}) < 0 \\ C2: & (\forall \mathbf{x} \not\in int(S))\ V(\mathbf{x}) > 0 \\ C3: & (\forall \mathbf{x} \in S \setminus int(T))(\exists \mathbf{u} \in U)\ \nabla V \cdot f(\mathbf{x},\mathbf{u}) < 0. \end{array} \tag{27}$$

We briefly sketch the argument as to why such a Lyapunov-like function satisfies the reach-while-stay, referring the reader to our earlier work on control certificates for a detailed proof [79]. Suppose we have found a function V satisfying (27). V is strictly negative over the initial set I and strictly positive outside the safe set S. Furthermore, as long as the flow remains inside the set S without reaching the interior of the target T, there exists a control input at each state to strictly decrease the value of V. Combining these observations, we conclude either (a) the flow remains forever inside set $S \setminus int(T)$ or (b) must visit the interior of set T (before possibly leaving S). However, option (a) is ruled out because $S \setminus int(T)$ is a compact set and V is a continuous function. Therefore, if the flow were to remain within $S \setminus int(T)$ forever then $V(\mathbf{x}(t)) \to -\infty$ as $t \to \infty$, which directly contradicts the fact that V must be lower bounded on a compact set $S \setminus int(T)$. We therefore, conclude that the flow must stay inside S and eventually visit the interior of the target T.

The learning framework extends easily to search for a function V that satisfies the constraints in Eq. (27).

6.4 Finite-time Reachability

The idea of funnels has been developed to use the Lyapunov argument for finite-time reachability [59]. Then, following Majumdar et al., a library of control funnels can provide building blocks for motion planning [57]. Likewise, control funnels are used to reduce reach-avoid problem to timed automata [15].

In this section, we consider Lyapunov-like functions for establishing control funnels. Let I be a set of initial states for the plant ($\mathbf{x}(0) \in I$), and T be the target set that the system should reach at time $\mathcal{H} > 0$ ($\mathbf{x}(\mathcal{H}) \in int(T)$). Let S be the safe set, such that $I, T \subseteq S$ and $\mathbf{x}(t) \in S$ for time $t \in [0,\mathcal{H}]$. The goal is to find a controller that guarantees that whenever $\mathbf{x}(0) \in I$, we have $\mathbf{x}(t) \in S$ for all $t \in [0,\mathcal{H}]$ and $\mathbf{x}(\mathcal{H}) \in int(T)$. To solve this, we search instead for a control Lyapunov-like function $V(\mathbf{x},t)$ that is a function of the state and time, with the following properties:

$$\begin{array}{ll} C1: & (\forall \mathbf{x} \in I)\ V(\mathbf{x},0) < 0 \\ C2: & (\forall \mathbf{x} \not\in int(T))\ V(\mathbf{x},\mathcal{H}) > 0 \\ C3: & \left(\forall \begin{array}{l} t \in [0,\mathcal{H}] \\ \mathbf{x} \not\in int(S) \end{array}\right) V(\mathbf{x},t) > 0 \\ C4: & \begin{pmatrix} \forall t \in [0,\mathcal{H}] \\ \forall \mathbf{x} \in S \end{pmatrix} (\exists \mathbf{u} \in U)\ \dot{V}(t,\mathbf{x},\mathbf{u}) < 0\,, \end{array} \tag{28}$$

where $\dot{V}(t,\mathbf{x},\mathbf{u}) = \frac{\partial V}{\partial t} + \nabla V \cdot f(\mathbf{x},\mathbf{u})$. First of all, when initialized to $\mathbf{x}(0) \in I$, we have $V(\mathbf{x},0) < 0$ by condition C1. Next, the controller's action through condition $C4$ guarantees that $\frac{dV}{dt} < 0$ over the trajectory for $t \in [0,\mathcal{H}]$, as long as $\mathbf{x} \in S$. Through $C3$, we can guarantee that $\mathbf{x}(t) \in S$ for $t \in [0,\mathcal{H}]$. Finally, it follows that $V(\mathbf{x}(\mathcal{H}),\mathcal{H}) < 0$. Through $C2$, we conclude that $\mathbf{x} \in int(T)$. As depicted in Fig. 8, the set $V^{=0}$ forms a barrier, and set $V^{<0}$ forms the required funnel, while $t \leq \mathcal{H}$.

Theorem 10 *Given compact semi-algebraic sets I, S, T, a time horizon $\mathcal{H}$, and a smooth function V satisfying Eq.* (28)*, there exists a control strategy s.t. for all traces of the closed loop system, if* $\mathbf{x}(0) \in I$*, then*

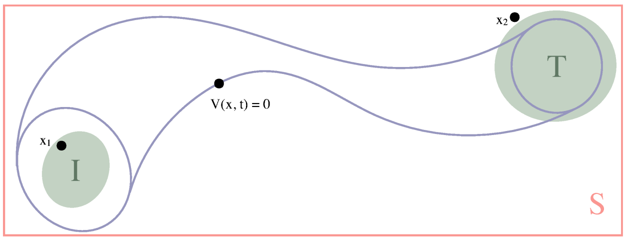


Fig. 8 A schematic view of a control funnel. Blue lines show the boundary of the funnel $V(\mathbf{x}, t) = 0$. Also, initially $V(\mathbf{x}_1, 0) < 0$ and at the end of horizon, $V(\mathbf{x}_2, \mathcal{H}) > 0$.

1. $(\forall t \in [0, \mathcal{H}])\ \mathbf{x}(t) \in S$
2. $\mathbf{x}(\mathcal{H}) \in int(T)$.

Proof Using Sontag result [89,102], there is a feedback $\mathcal{K}$ which decreases value of V while $t \in [0, \mathcal{H}]$ and $\mathbf{x} \in S$:

$$(\forall t \in [0, \mathcal{H}], \mathbf{x} \in S)\ \dot{V}(t, \mathbf{x}, \mathcal{K}(\mathbf{x})) < 0\,.$$

Now, assume $\mathbf{x}(0) \in I$. By the first condition of Eq. (28), $V(\mathbf{x}(0), 0) < 0$. Assume there is a time $t \in [0, \mathcal{H}]$ s.t. $\mathbf{x}(t) \notin S$. By compactness of S, and smooth dynamics, there is a time t_2 s.t. $V(\mathbf{x}(t_2), t_2) \in \partial S$ and for all $t < t_2$, $\mathbf{x}(t) \in int(S)$. According to the third condition of Eq. (28), $V(\mathbf{x}(t_2), t_2) > 0$. Since V is a smooth function there is a time t_1 $(0 < t_1 < t_2)$ s.t. $V(\mathbf{x}(t_1), t_1) = 0$ and for all $t < t_1$, $V(\mathbf{x}(t), t) \in S$. By the fourth condition in Eq. (28):

$$\begin{aligned} V(\mathbf{x}(t_1), t_1) &= V(\mathbf{x}(0), 0) + \int_0^{t_1} \dot{V}(t, \mathbf{x}(t), \mathcal{K}(\mathbf{x}(t))) \\ &< V(\mathbf{x}(0), 0) < 0\,. \end{aligned}$$

This is a contradiction and therefore, for all $t \in [0, \mathcal{H}]$, $\mathbf{x}(t) \in S$. And similar to the argument above, it is guaranteed that for all $t \in [0, \mathcal{H}]$, $V(\mathbf{x}(t), t) < 0$. By the second condition of Eq. (28), it is guaranteed that if $\mathbf{x}(\mathcal{H}) \notin int(T)$, then $V(\mathbf{x}(\mathcal{H}), \mathcal{H}) > 0$. Therefore, $\mathbf{x}(\mathcal{H}) \in int(T)$.

Using the Lyapunov-like conditions (28), the problem of finding such control funnels (respecting Eq. (28)) belongs to the class of problem which could be solved with our method.

7 Experiments

In this section, we describe numerical results on some case studies. We first describe our implementation of the techniques described thus far. The verifier component is implemented using tool Gloptipoly [34], which in turn uses Mosek to solve SDP problems [61], and only needs a degree of relaxation D as its input. For the demonstrator, a nonlinear MPC scheme is used, which is solved using a gradient descent algorithm. For each benchmark, the following parameters are tuned to obtain the cost function:

1. time step τ
2. number of horizon steps N
3. Q, R, and H for the cost function:

$$\begin{aligned} &\left(\textstyle\sum_{i=1}^{N-1} \mathbf{x}(i\tau)^t\ Q\ \mathbf{x}(i\tau) + \mathbf{u}(i\tau)^t\ R\ \mathbf{u}(i\tau)\right) \\ &\qquad + \mathbf{x}(N\tau)^t\ H\ \mathbf{x}(N\tau)\,. \end{aligned}$$

As such, an MPC cost function is designed to enforce a specification such as stability or reaching a target set. However, since the approach provides no guarantees, we run hundreds of simulations of the closed loop system starting from randomly selected initial states to check whether the specifications are met. Failing this, the cost function is adjusted, repeating the testing process. And finally, for the learner, quadratic polynomials are used as candidates for the desired Lyapunov-like functions. Nevertheless, more complicated polynomials are also supported by our implementation. Beside these inputs, each control problem has a specification. For example, for a *reach-while-stay* problem, the target set T, initial set I, and safe set S are provided as inputs.

All the computations reported in this section were performed on a Mac Book Pro with 2.9 GHz Intel Core i7 processor and 16GB of RAM. The reported CLFs are rounded to two decimal points. The implementation is available upon request.

7.1 Case Study I:

This system is two-wheeled mobile robot modeled with five states $[x, y, v, \theta, \gamma]$ and two control inputs [27], where x and y define the position of the robot, v is its velocity, θ is the rotational position and γ is the angle between the front and rear axles. The goal is to stabilize the robot to a target velocity $v^* = 5$, and $\theta^* = \gamma^* = y^* = 0$ as shown in Fig. 9. The dynamics of the model is as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \\ \dot{\sigma} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ u_1 \\ v\sigma \\ u_2 \end{bmatrix},$$

where $\sigma = tan(\gamma)$ (see Fig. 9). Variable x is immaterial in the stabilization problem and is dropped to obtain a model with four state variables $[y, v, \theta, \sigma]$. Also, the sine function is approximated with a polynomial of degree one. The inputs are saturated over the intervals

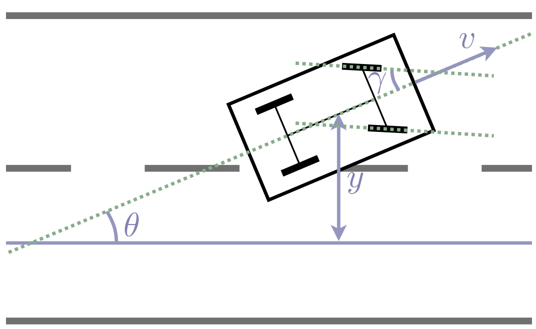


Fig. 9 A schematic view of the bicycle model.

$U : [-10, 10] \times [-10, 10]$, and the specification is reach-while-stay, provided by the following sets

$$S : [-2, 2] \times [3, 7] \times [-1, 1] \times [-1, 1]$$
$$I : \mathcal{B}_{0.4}(\mathbf{0})$$
$$T : \mathcal{B}_{0.1}(\mathbf{0})\,.$$

The method finds the following CLF:

$$V = 0.37y^2 + 0.52y\theta + 3.11\theta^2 + 0.98y\sigma + 2.23\sigma\theta + 4.46\sigma^2 - 0.36vy - 0.29v\theta + 0.95v\sigma + 3.86v^2\,.$$

This CLF is used to design a controller. Fig. 10 shows the projection of trajectories on to x-y plane for the synthesized controller in red. The blue trajectories are generated using the MPC controller that served as the demonstrator. The behavior of the system for both controllers are similar but not identical. Notice that the initial state in Fig. 10(c) is not in the region of attraction (guaranteed region). Nevertheless, the CLF-based controller can still stabilize the system while keeping the system in the safe region. On the other hand, the MPC violates the safety constraints even when the safety constraints are imposed in the MPC scheme. The safety is violated because in the beginning θ gets larger than 1 and it gets close to $\pi/2$ (the robots moves almost vertically).

7.2 Case Study II:

The problem of keeping the inverted pendulum in a vertical position is considered. This case study has applications in balancing two-wheeled robots [20]. The system has two degrees of freedom: the position of the cart x, and the degree of the inverted pendulum θ. The goal is to keep the pendulum in a vertical position by moving the cart with input u (Fig. 11).

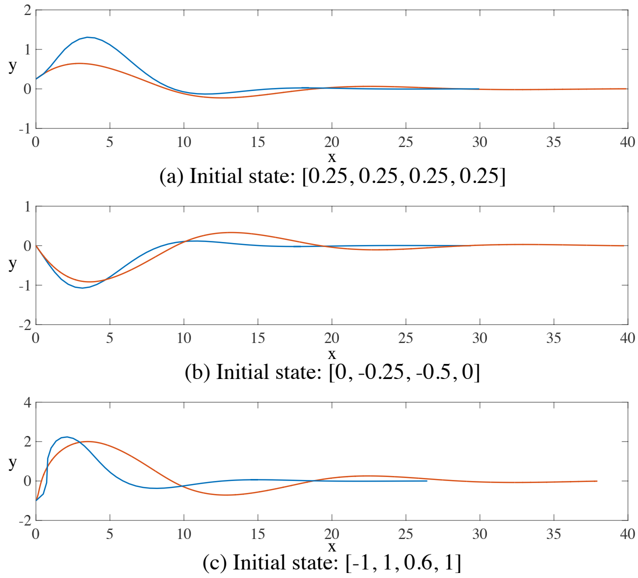


Fig. 10 Simulation for the bicycle robot - Projected on x-y plane. Simulation traces are plotted for three different initial states. Blue (red) traces corresponds to trajectories of the system for MPC controller (CLF-based controller).

The system has four state variables $[x, \dot{x}, \theta, \dot{\theta}]$ with the following dynamics [46]:

$$\begin{bmatrix} \ddot{\mathbf{x}} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{4u - 4\epsilon\dot{x} + 4ml\dot{\theta}^2\sin(\theta) - 3mg\sin(\theta)\cos(\theta)}{4(M+m) - 3m\cos^2(\theta)} \\ \frac{(M+m)g\sin(\theta) - (u - \epsilon\dot{x})\cos(\theta) - ml\dot{\theta}^2\sin(\theta)\cos(\theta)}{l(\frac{4}{3}(M+m) - m\cos(\theta)^2)} \end{bmatrix},$$

where $m = 0.21$ and $M = 0.815$ are masses of the pendulum and the cart respectively, $g = 9.8$ is the gravitational acceleration, and $l = 0.305$ is distance of center of mass of the pendulum from the cart. After partial linearization, the dynamics have the following form:

$$\begin{bmatrix} \ddot{\mathbf{x}} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 4u + \frac{4(M+m)g\tan(\theta) - 3mg\sin(\theta)\cos(\theta)}{4(M+m) - 3m\cos^2(\theta)} \\ \frac{-3u\cos(\theta)}{l} \end{bmatrix}.$$

The trigonometric and rational functions are approximated with polynomials of degree three. The input is saturated $U : [-20, 20]$ and sets for a safety specification are $S : [-1, 1]^4$, $I : \mathcal{B}_{0.1}(\mathbf{0})$.

Fig. 12 shows the some of the traces of the closed loop system for the CLF-based controller as well as the MPC controller. Notice that the trajectories of the CLF based controller are quite distinct from the MPC, especially in regions where the demonstration is not provided during the CLF synthesis process. For example, in Figure. 12(b), the behaviors of these controllers are similar outside the initial set I. However, inside I (near the equilibrium) the behavior is different, since the demonstrations are only generated for states outside I. The CLF-based controller is designed using the following CLF generated by the learning framework:

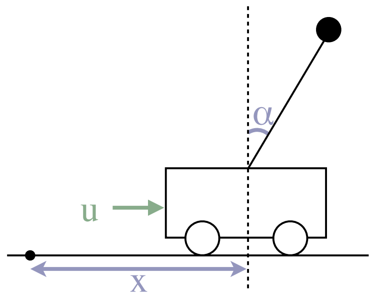


Fig. 11 A schematic view of the "inverted pendulum on a cart".

$$V = 16.37\dot{\theta}^2 + 50.37\dot{\theta}\theta + 75.16\theta^2 + 13.51x\dot{\theta} + 43.26x\theta + 10.44x^2 + 23.30\dot{\theta}\dot{x} + 38.09\dot{x}\theta + 11.13\dot{x}x + 9.55\dot{x}^2 .$$

7.3 Case Study III:

Caltech ducted fan has been used to study the aerodynamics of a single wing of a thrust vectored, fixed wing aircraft [37]. In this case study, we wish to design forward flight control in which the angle of attack needs to be set for a stable forward flight. The model of the system is carefully calibrated through wind tunnel experiments. The system has four states: v is the velocity, γ is the moving direction the ducted fan, θ is the rotational position, and q is the angular velocity. The control inputs are the thrust u and the angle at which the thrust is applied δ_u (Fig. 13). Also, the inputs are saturated: $U : [0, 13.5] \times [-0.45, 0.45]$. The dynamics are:

$$\begin{bmatrix} m\dot{v} \\ mv\dot{\gamma} \\ \dot{\theta} \\ J\dot{q} \end{bmatrix} = \begin{bmatrix} -D(v,\alpha) - W\sin(\gamma) + u\cos(\alpha + \delta_u) \\ L(v,\alpha) - W\cos(\gamma) + u\sin(\alpha+\delta_u) \\ q \\ M(v,\alpha) - ul_T\sin(\delta_u) \end{bmatrix},$$

where the angle of attack $\alpha = \theta - \gamma$, and D, L, and M are polynomials in v and α. For full list of parameters, see [37]. According to the dynamics, $\mathbf{x}^* : [6, 0, 0.1771, 0]$ is a stable equilibrium (for $\mathbf{u}^* : [3.2, -0.138]$) where the ducted fan can move forward with velocity 6. Thus, the goal is to reach near $\mathbf{x}^*$. The system is not affine in control. We replace u and δ_u with $u_s = u\sin(\delta_u)$ and $u_c = u\cos(\delta_u)$:

$$\begin{bmatrix} \dot{v} \\ \dot{\gamma} \\ \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \frac{-D(v,\alpha) - W\sin(\gamma) + u_c\cos(\alpha) - u_s\sin(\alpha)}{m} \\ \frac{L(v,\alpha) - W\cos(\gamma) + u_c\sin(\alpha) + u_s\cos(\alpha)}{mv} \\ q \\ \frac{M(v,\alpha) - l_T u_s}{J} \end{bmatrix}.$$

Projection of U into the new coordinate will yield a sector of a circle. Then, set U is safely under-approximated by a polytope $\hat{U}$ as shown in Fig. 14. Next, we perform a translation so that the $\mathbf{x}^*$ ($\mathbf{u}^*$) is the origin of the state (input) space in the new coordinate system. In order to obtain a polynomial dynamics, we approximate v^{-1}, sin and cos with polynomials of degree one, three and three, respectively. These changes yield a polynomial control affine dynamics, which fits the description of our model. For the reach-while-stay specification, the sets are defined as the following:

$$S : [3, 9] \times [-0.75, 0.75] \times [-0.75, 0.75] \times [-2, 2]$$
$$I : \{[v, \gamma, \theta, q]^t | (0.4v)^2 + \gamma^2 + \theta^2 + q^2 < 0.4^2\}$$
$$T : \{[v, \gamma, \theta, q]^t | (0.4v)^2 + \gamma^2 + \theta^2 + q^2 < 0.05^2\} .$$

The projection of some of the traces of the system in x-y plane is shown in Fig. 15. We set $x_0 = y_0 = 0$ and

$$\dot{x} = v\cos(\gamma),\ \dot{y} = v\sin(\gamma) .$$

The CLF-based controller is designed using the following generated CLF:

$$V = +3.23q^2 + 2.17q\theta + 3.90\theta^2 - 0.2qv - 0.45v\theta + 0.53v^2 + 1.66q\gamma - 1.33\gamma\theta + 0.48v\gamma + 3.90\gamma^2 .$$

The traces show that the CLF-based controller stabilizes faster, however, the MPC controller uses the aerodynamics to achieve the same goal with a better performance.

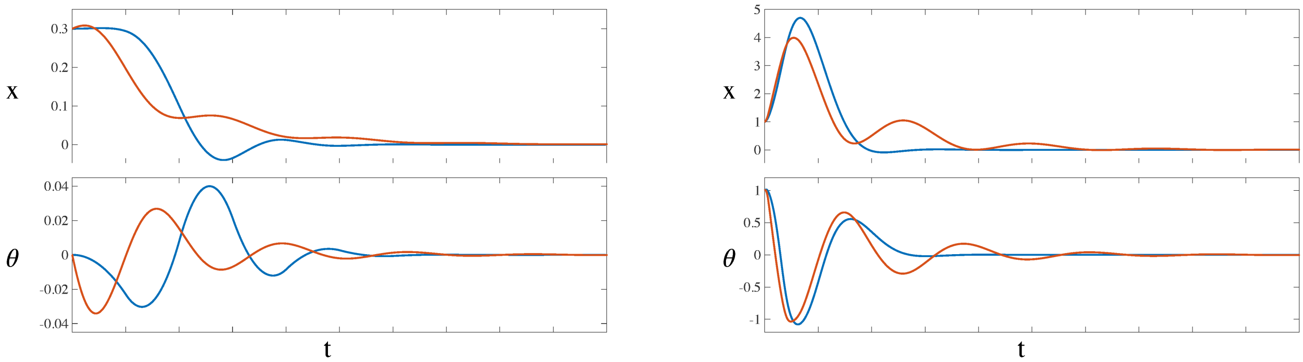
7.4 Case Study IV:

This case study addresses another problem for the planar Caltech ducted fan [37]. The goal is to keep the planar ducted fan in a hover mode. The system has three degrees of freedom, x, y, and θ, which define the position and orientation of the ducted fan. There are six state variables x, y, θ, $\dot{x}$, $\dot{y}$, $\dot{\theta}$ and two control inputs u_1, u_2 ($U \in [-10, 10] \times [0, 10]$). The dynamics are

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ J\ddot{\theta} \end{bmatrix} = \begin{bmatrix} -d_c\dot{x} + u_1\cos(\theta) - u_2\sin(\theta) \\ -d_c\dot{y} + u_2\cos(\theta) + u_1\sin(\theta) - mg \\ ru_1 \end{bmatrix},$$

where $m = 11.2$, $g = 0.28$, $J = 0.0462$, $r = 0.156$ and $d_c = 0.1$. The system is stable at origin for $\mathbf{u}^* : [0, mg]$. Therefore, we set $\mathbf{u}*$ as the origin for the input space. The specification is a reach-while-stay property with the following sets:

$$S : [-1, 1] \times [-1, 1] \times [-0.7, 0.7] \times [-1, 1]^3$$
$$I : \mathcal{B}_{0.25}(\mathbf{0}), T : \mathcal{B}_{0.1}(\mathbf{0}) .$$



(a) Initial state $(0.3, 0, 0, 0)$ (b) Initial state $(1, 1, 1, 1)$

Fig. 12 Simulation for the inverted pendulum system. Simulation traces are plotted for two initial states. Red (blue) traces show the simulation for the CLF-based (MPC) controller.

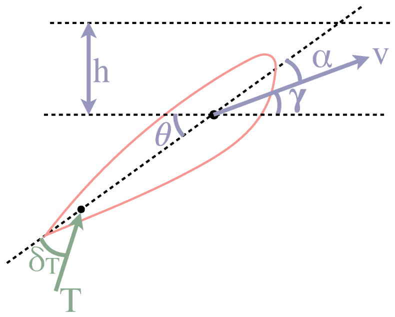


Fig. 13 A schematic view of the Caltech ducted fan.

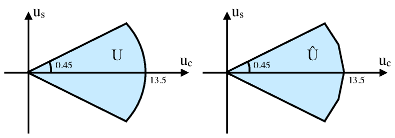


Fig. 14 Set of feasible inputs U and its under approximation $\hat{U}$ in the new coordinate for case study III.

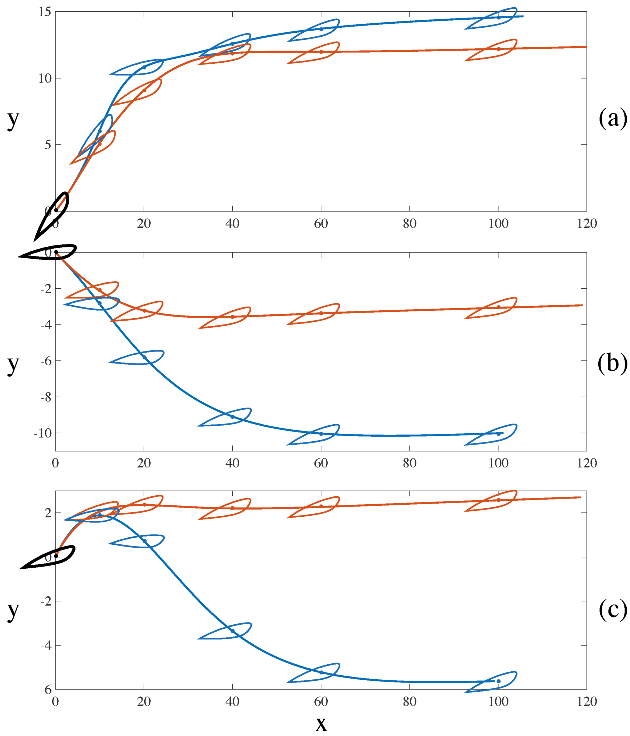
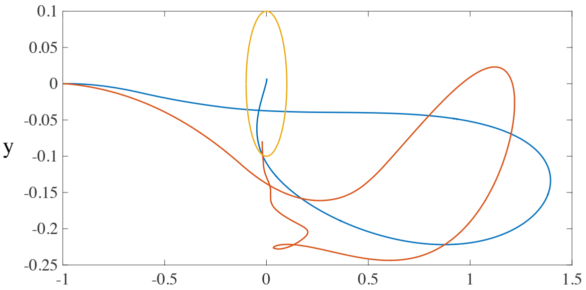


Fig. 15 Simulation for forward flight of Caltech ducted fan - Projected on x-y plane. Blue (red) traces are trajectories of the closed loop system with the MPC (CLF-based) controller. The rotational position is shown for some of the states (in black for the initial state) for each trajectory. Initial states are $[2, 0.4, 0.717, 0]$, $[-1, -0.25, -0.133, 0]$, and $[-1, 0.4, 0.177, 0]$ for (a), (b), and (c), respectively.

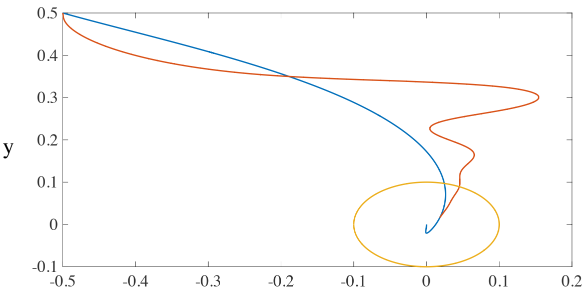
The trigonometric functions are approximated with degree two polynomials and the procedure finds a quadratic CLF:

$$\begin{aligned}V =&1.64\dot{\theta}^2 - 0.56\dot{\theta}\dot{y} + 13.53\dot{y}^2 + 0.07\dot{\theta}y + 1.15y\dot{y}+\\ &1.16y^2 + 1.74\theta\dot{\theta} + 0.03\dot{y}\theta - 0.77y\theta + 4.80\theta^2-\\ &4.57\dot{\theta}\dot{x} + 0.85\dot{x}\dot{y} + 0.34y\dot{x} - 8.59\dot{x}\theta + 12.77\dot{x}^2-\\ &0.45\dot{\theta}x + 0.06\dot{y}x + 0.51yx - 3.71x\theta + 4.12x\dot{x}+\\ &1.88x^2\,.\end{aligned}$$

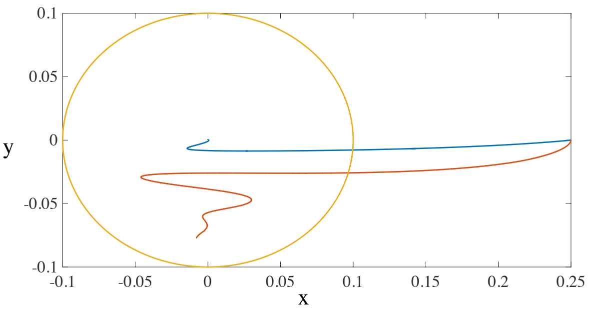
Some of the traces are shown in Fig. 16. As the simulation suggest, the MPC controller behaves very differently and the CLF-based controller yield solutions with more oscillations. The CLF-based controller first stabilizes x and θ and then value of y settles. Also, once the trace is inside the target region, the CLF-based controller does not guarantee decrease in V as this fact is intuitively visible in Fig. 16(c).



(a) Initial state: [-1, 0, -0.7, 1, 0, 0]



(b) Initial state: [-0.5, 0.5, -0.5, 0, 0, 0]



(c) Initial state: [0.25, 0, 0.25, 0, 0, 0]

Fig. 16 Simulation for Case Study IV - Projected on x-y plane. The trajectories corresponding to the CLF-based (MPC) controller are shown in red (blue) lines. The boundary of the target set is shown in yellow.

7.5 Case Study V:

In this case study, a unicycle model [52] is considered. It is known that no continuous feedback can stabilize the unicycle, and therefore no continuous CLF exists. However, considering a reference trajectory for a moving unicycle, one can keep the system near the reference trajectory, using control funnels. The unicycle model has the dynamics:

$$\dot{x} = u_1 \cos(\theta) \ , \ \dot{y} = u_1 \sin(\theta) \ , \ \dot{\theta} = u_2 \, .$$

By a change of basis, a simpler dynamic model is used here (see. [52]):

$$\dot{x_1} = u_1, \dot{x_2} = u_2, \dot{x_3} = x_1 u_2 - x_2 u_1 \, .$$

We consider a planning problem, in which starting near $[\theta, x, y] = [\frac{\pi}{2}, -1, -1]$, the goal is to reach near $[\theta, x, y] = [0, 2, 0]$. In the first step, a feasible trajectory $\mathbf{x}^*(t)$ is generated as shown in Fig. 17(a). Then $\mathbf{x}^*(t)$ is approximated with piecewise polynomials. More precisely, trajectory consists of two segments. The first segment brings the car to the origin and the second segment moves the car to the destination. Each segment is approximated using polynomials in t with degree up to three:

$$\text{seg. } 2: \begin{cases} \theta(t)^* = 0 \\ x^*(t) = t \\ y^*(t) = 0 \end{cases}$$

$$\text{seg. } 1: \begin{cases} \theta^*(t) = \pi - t \\ x^*(t) = -(1 - 0.64t)(1 + 0.64t) \\ y^*(t) = -(1 - 0.64t)(1 - 0.2t - 0.25t^2) \, . \end{cases}$$

Let $Tr(\theta, x, y)$ represent the transformation of the state in terms of (θ, x, y) coordinate system to the (x_1, x_2, x_3) coordinates. Also, for two set A, and B, let $A \oplus B$ be the Minkowski sum of A and B. For example, we write $\{Tr(\theta, x, y)\} \oplus \mathcal{B}_\delta(\mathbf{0})$ to denote a state and a ball of radius δ around it. Moreover, let S_1 (S_2) be the minimal box which contains the trajectory $\mathbf{x}^*(\cdot)$ for the first (second) segment in the (x_1, x_2, x_3) coordinates. For the first segment, the goal is to reach from the initial set $I : \{Tr(\pi/2, -1, -1)\} \oplus \mathcal{B}_1(\mathbf{0})$ to the target set $T : \{Tr(0, 0, 0)\} \oplus \mathcal{B}_1(\mathbf{0})$. Also, the safe set is defined as $S : S_1 \oplus [-1.5, 1.5]^3$. That is, an enlarged box around S_1. And in the next segment, the goal is to reach from initial set $I : Tr(0, 0, 0) \oplus \mathcal{B}_1(\mathbf{0})$ to $T : Tr(0, 2, 0) \oplus \mathcal{B}_1(\mathbf{0})$ as the target, while staying in $S : S_2 \oplus [-2, 2]^3$.

For each segment, we search for a Lyapunov-like function V as a time varying function, quadratic in the states. Our method is applied to this problem, and we are able to find a strategy to implement the plan with guarantees. The boundary of the funnels is shown in Fig. 17(a). Also, some simulation traces are shown in Fig. 17(b), where the CLF controller is implemented using the generated funnels. As simulations suggest, the funnels can effectively stabilize the traces to the trajectory, when the unicycle is moving forward.

7.6 Performance

As mentioned earlier, the inputs to the learning framework are the plant, monomial basis functions, and the demonstrator. Also, the degree of relaxation D is also considered as input. At each iteration, first a MVE inscribed inside a polytope is calculated. This task is performed quite efficiently. The MPC scheme used inside

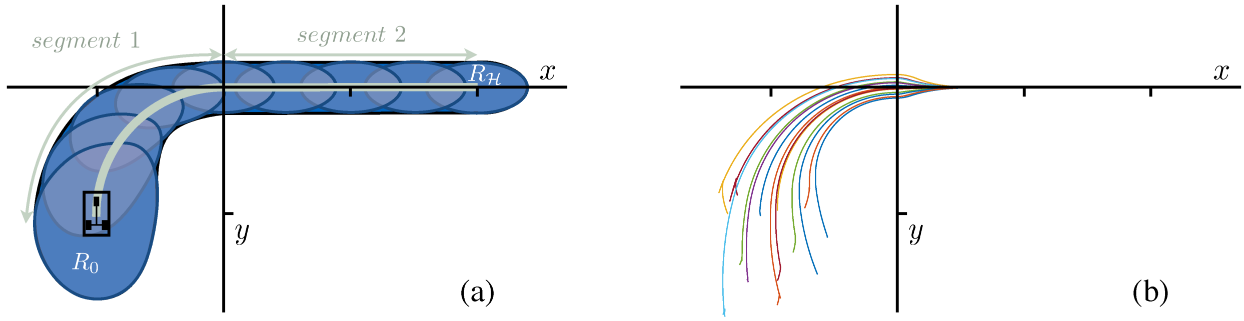


Fig. 17 (a) Trajectory tracking using control funnel - Projected on x-y plane. The reference trajectory is shown with the green line, consists of two segments. Starting from R_0, the state remains in the funnel (blue region) until it reaches $R_{\mathcal{H}}$. Boundary of each smaller blue region shows the boundary of the funnel for a specific time. (b) Simulation traces for some random initial states.

the demonstrator is an input and we do not consider its performance here. Nevertheless, MPC is known to be very efficient if it is carefully tuned. We mention that the MPC parameters used here are selected by a non-expert and usually the time step is very small and the horizon is very long. Nevertheless, as the MPC is used offline, they are still suitable for our framework. Also, costs matrices Q, R, and H are diagonal:

$$Q = diag(Q') \ , \ R = diag(R') \ , \ H = N diag(Q') \, ,$$

where $Q' \in \mathbb{R}^n$ and $R' \in \mathbb{R}^m$. There are two other important factors that determines the performance of the whole learning framework: (i) the time taken by the verifier and (ii) the number of iterations. Table. 1 shows the results of the learning framework for the set of case studies described thus far. For each problem instance, the parameters of the MPC, as well as the degree of relaxation are provided. Also, the performance of the learning framework is tabulated. First, the procedure starts from $C : [-\Delta, \Delta]^r$ and terminates whenever $\mathrm{Vol}(E_j) < \gamma\delta^r$. We set $\Delta = 100$ and $\delta = 10^{-3}$. The results demonstrate that the method terminates in few iterations, even for the cases where a compatible CLF does not exists.

Notice that the number of demonstrations is different from the number of iterations. Recall that two separate problems are solved for the verification. One involves checking the positivity of V, and the other involves checking whether ∇V can be decreased. When a counterexample $\mathbf{x}_j$ is found for the former problem, there is no need to check the latter condition. Furthermore, we do not require a demonstration for such a scenario. This optimization is added to speed up our overall procedure by avoiding expensive calls to the MPC. To accommodate this, our approach calculates $\hat{C}_{j+1}$ (instead of C_{j+1}) for such counterexamples as:

$$\hat{C}_{j+1} : \ \hat{C}_j \cap \{\mathbf{c} \mid V_{\mathbf{c}}(\mathbf{x}_j) > 0\} \ . \tag{29}$$

Otherwise, if the counterexample violates conditions on ∇V, then

$$\hat{C}_{j+1} : \ \hat{C}_j \cap \left\{ \mathbf{c} \mid \begin{array}{c} V_{\mathbf{c}}(\mathbf{x}_j) > 0 \\ \nabla V_{\mathbf{c}}.f(\mathbf{x}_j, \mathbf{u}_j) < 0 \end{array} \right\} . \tag{30}$$

However, $\mathbf{c}_j \notin \hat{C}_{j+1}$ for both cases and the convergence guarantees continue to hold. As Table. 1 shows, using this trick, the number of demonstrations can be much smaller than the total number of iterations.

At each iteration, several verification problems are solved which involve solving large SDP problems. While the complexity of solving SDP is polynomial in the number of variables, they are still hard to solve. The verification problem is quite expensive when the number of variables and degree of relaxation are large. Nevertheless, as the SDP solvers mature further, we believe our method can solve larger problems, since the verification procedure is currently the computational bottleneck for the learning framework. We note that, using larger degree of relaxation does not necessarily lead to a longer learning process (e.g. hover flight example). For example, for the inverted pendulum example, using degree of relaxation five the procedure finds a CLF faster when compared to the case wherein the degree of relaxation is set to four.

In previous sections, we discussed that two important factor governs the convergence of the search process: (i) candidate selection, and (ii) counterexample selection. In order to study the effect of these processes, we investigate different techniques to evaluate their performances. For candidate selection, we consider three different methods. In the first method, a Chebyshev center of C_j is used as a candidate. In the second method, the analytic center of constraints defining C_j is the selected candidate and redundant constraints are not dropped. And finally, in the last method, the center of MVE inscribed in C_j yields the candidate. Also, for each of these methods, we compare the performance for

Table 1 Results on the benchmark. τ: MPC time step, N: number of horizon steps, Q': defines MPC state cost, R': defines MPC input cost, D: SDP relaxation degree bound, #Dem : number of demonstrations, #Itr: number of iterations, V. Time: total computation time for verification (minutes), Time: total computation time (minutes)

Problem	Demonstrator				Verifier	Performance				
System Name	τ	N	Q'	R'	D	#Dem	# Itr	V. Time	Time	Status
Unicycle-Segment 2	0.1	10	[1 1 1]	[1 1]	3	2	74	3	3	Fail
					4	2	57	4	4	Succ
Unicycle-Segment 1	0.1	20	[1 1 1]	[1 1]	3	27	86	9	10	Fail
					4	23	71	11	12	Succ
TORA	1	30	[1 1 1 1]	[1]	3	52	118	7	14	Fail
					4	19	76	5	8	Succ
Inverted Pendulum	0.04	50	[10 1 1 1]	[10]	3	56	85	7	27	Fail
					4	53	69	9	25	Succ
					5	34	50	7	19	Succ
Bicycle	0.4	20	[1 1 1 1]	[1 1]	2	14	32	2	2	Fail
					3	7	25	1	1	Succ
Bicycle × 2	0.4	20	[1 1 1 1 1 1 1 1]	[1 1 1 1]	2	119	225	77	90	Fail
					3	30	81	43	46	Succ
Forward Flight	0.4	40	[1 1 1 1]	[1 1]	4	14	77	16	18	Fail
					5	4	64	10	10	Succ
Hover Flight	0.4	40	[1 1 1 1 1 1]	[1 1]	2	57	147	12	40	Fail
					3	57	124	21	47	Succ
					4	51	116	30	54	Succ

two different cases: (i) a random counterexample is generated, (ii) the generated counterexample maximizes constraint violations (see Sec. 5.3). Table 2 shows the performance for each of these cases, applied to the same set of problems. The results demonstrate that selecting good counterexamples would increase the convergence rate (fewer iterations). Nevertheless, the time it takes to generate these counterexamples increases, and therefore, the overall performance degrades. In conclusion, while generating good counterexamples provides better reduction in the space of candidates, it is computationally expensive, and thus, it seems to be beneficial to just rely on candidate selection for fast termination. Table. 2 also suggests that Chebyshev center has the worst performance. Also, the MVE-based method performs better (fewer iterations) compared to the method which is based on the analytic center.

7.7 Comparison with Other Approaches

We now compare our method against other techniques used to automatically construct provably correct controllers.

Comparison with CEGIS: We have claimed that the use of demonstrator helps our approach deal with a computationally expensive quantifier alternation in the CLF condition. To understand the impact of this aspect of our approach, we first we compare the proposed method with our previous work, namely counterexample guided inductive synthesis (CEGIS) that is designed to solve constraints with quantifier alternation, and applied to the synthesis of CLFs [78]. In this framework, the learning process only relies on counterexamples provided by a verifier component, without involving demonstrations. Despite a timeout that is set to two hours, our CEGIS method timed out for *all the problem instances* discussed in this article, without discovering a CLF. As a result, we exclude this approach from further comparisons. These results suggest that demonstrations are essential for fast convergence.

Learning CLFs from Data: On the other hand, Khansari-Zadeh et al. [83] learn likely CLFs from demonstrations from sets of states that are sampled without (a) the use of a verifier to check, and (b) counterexamples as new samples, both of which are features of our approach. Therefore, the correctness of the controller thus derived is not formally guaranteed. To this end, we verify if the solution is in fact a CLF.

The methodology of Khansari-Zadeh et al. is implemented using the following steps:

1. Choose a parameterization of the desired CLF $V_{\mathbf{c}}(\mathbf{x})$ (identical to our approach).
2. Generate samples in batches, wherein for each batch:
   (a) Sample $N_1 = 50$ states uniformly at random, and for each state $\mathbf{x}_i$, add the constraint $V_{\mathbf{c}}(\mathbf{x}_i) \geq 0$, for $i \in [1, N_1]$.
   (b) Sample $N_2 = 5$ States at random, and for each state $\mathbf{x}_j$ ($j \in [1, N_2]$), simulate the MPC demonstrator for $N_3 = 10$ time steps to obtain state control samples

$$\{(\mathbf{x}_{j,1}, \mathbf{u}_{j,1}), \ldots, (\mathbf{x}_{j,N_3}, \mathbf{u}_{j,N_3})\}\,.$$

Table 2 Results on different variations. I: number of iterations, VT: computation time for verification (minutes), T: total computation time (minutes), Simple CE: any counterexample, Max CE: counterexample with maximum violation

Problem	Chebyshev Center						Analytic Center						MVE Center					
	Simple CE			Max CE			Simple CE			Max CE			Simple CE			Max CE		
	I	VT	T	I	VT	T	I	VT	T	I	VT	T	I	VT	T	I	VT	T
Unicycle - Seg. 2	83	4	4	22	9	9	76	5	6	23	9	10	57	4	4	15	6	6
Unicycle - Seg. 1	81	6	7	34	17	17	85	10	10	35	15	16	71	11	12	36	18	18
TORA	185	7	10	52	12	15	95	5	9	36	9	11	76	5	8	36	12	14
Inverted Pend.	163	10	23	85	22	30	57	8	20	51	22	32	50	7	19	35	18	25
Bicycle	99	3	3	40	5	5	31	2	2	20	3	3	25	1	2	15	3	3
Bicycle $\times$ 2	759	121	127	438	244	246	96	47	50	77	141	143	81	43	46	66	132	133
Forward Flight	676	20	21	34	30	31	113	15	16	21	18	19	64	10	10	16	16	16
Hover Flight	499	65	90	196	113	127	146	36	67	90	92	109	116	30	54	75	69	82

(c) Add the constraints $\nabla V_{\mathbf{c}} \cdot f|_{\mathbf{x}=\mathbf{x}_{j,k}, \mathbf{u}=\mathbf{u}_{j,k}} < 0$ for $j = 1, \ldots, N_2$ and $k = 1, \ldots, N_3$ to enforce the negative definiteness of the CLF.

3. At the end of batch k, solve the system of linear constraints thus far to check if there is a feasible solution.
4. If there is no feasible solution, then **exit**, since no function in $V_{\mathbf{c}}(\mathbf{x})$ is compatible with the data.
5. If there is a feasible solution, check this solution using the VERIFIER.
6. If the verifier succeeds, then **exit** successfully with the CLF discovered.
7. Otherwise, continue to generate another batch of samples.

We enforce the constraint $V(\mathbf{x}) > 0$ and $\nabla V \cdot f < 0$ over different sets of samples, since simulating the demonstrator is much more expensive for each point. The approach iterates between generating successive batches of data until a preset timeout of two hours as long as (a) there are CLFs remaining to consider and (b) no CLF has been discovered thus far. The time taken to learn and verify the solution is not considered against the total time limit, and also not added to the overall time reported. Besides stability, the approach is also adapted for other properties, which are used in our benchmarks.

The results are reported in Table. 3. Since the generation of random samples are involved, we run the procedure 10 times on each benchmark, and report the percentage of trials that succeeded in finding a CLF, the number of timeouts and the number of trials that ended in an infeasible set of constraints. We note that the success rate is 100% for just one problem instance. For four other problem instances, the method is successful for a fraction of the trials. The remaining benchmarks fail on all trials. Next, the minimum and maximum number of demonstrations needed in the trials to find a CLF is reported as the "best-case" and "worst-case" respectively. We note that our approach requires much fewer demonstrations even when compared the best case scenario. Thus, we conclude from this data that the time spent by our approach for finding counterexamples is justified by the *significant decrease* in the number of demonstrations, and thus, faster convergence. This is beneficial especially for cases where generating demonstrations is expensive.

For one of the benchmarks (the forward flight problem of the Caltech ducted fan), the method stops for all cases because a function compatible with the data does not exist. As such, this suggests that no CLF compatible with the demonstrator exists. On the other hand, our approach successfully finds a CLF while considering just four demonstrations.

Finally, for two of the larger problem instances, we continue to obtain feasible solutions at the end of the time limit, although the verifier cannot prove the learned function is a CLF. In other words, there are values of $\mathbf{c}$ left, that have not been considered by the verifier. Our approach uses counterexamples, along with a judicious choice of candidate CLFs to eliminate all but a bounded volume of candidates.

Comparison with Bilinear Solvers: We now compare our method against approaches based on bilinear formulations found in related work [25,56,94]. We wish to find a Lyapunov function V and a corresponding feedback law $K : X \mapsto U$, simultaneously. Therefore, we assume K is a linear combination of basis functions $K : \sum_{k=1}^{r'} \theta_k h_k(\mathbf{x})$. Likewise, we parameterize V as a linear combination of basis functions, as well: $V : \sum_{k=1}^{r} c_k g_k(\mathbf{x})$. Then, we wish to find $\mathbf{c}$ and θ that satisfy the constraints corresponding to the property at hand. To synthesize a CLF, we wish to find $V_{\mathbf{c}}, K_\theta$, so that $V_{\mathbf{c}}(\mathbf{x})$ and its Lie derivative under the feedback $u = K_\theta(\mathbf{x})$ is negative definite. This is relaxed as an optimization problem:

$$\begin{aligned} \min_{\mathbf{c},\theta,\gamma}\ & \gamma \\ \text{s.t.}\ & V_{\mathbf{c}} \text{ is positive definite} \\ & (\forall\, \mathbf{x} \neq \mathbf{0})\ \nabla V_{\mathbf{c}}(\mathbf{x}) \cdot f(\mathbf{x}, K_\theta(\mathbf{x})) \leq \gamma ||\mathbf{x}||_2^2 \end{aligned}$$

The decision variables include $\mathbf{c}, \theta$ that parameterize V and K, respectively. In fact, if a feasible solution is obtained such that $\gamma < 0$ then we may stop the

Table 3 Results for "demonstration-only" method. #Sam.: number of samples, #Dem: number of demonstrations, Case: best-case or worst-case, Time: total computation time (minutes), TO: time out (> 2 hours).

Problem	Stats		Performance					Proposed Method		
System Name	Succ. %	TO %	Case	#Sam.	#Dem.	Time	Status	#Sam.	#Dem.	Time
Unicycle-Segment 2	60	0	best worst	400 600	40 72	1 1	Succ Fail	65	2	4
Unicycle-Segment 1	45	0	best worst	600 800	35 70	2 3	Succ Fail	79	23	12
TORA	60	30	best worst	6300 17100	535 1580	43 TO	Succ Fail	84	19	8
Inverted Pendulum	30	70	best worst	2250 15750	137 300	84 TO	Succ Fail	58	34	19
Bicycle	100	0	best worst	2700 54000	55 1883	2 48	Succ Succ	33	7	1
Bicycle × 2	0	100	best worst	81600 -	1736 -	TO -	Fail -	89	30	46
Forward Flight	0	0	best worst	900 2700	35 254	4 31	Fail Fail	72	4	10
Hover Flight	0	100	best worst	7150 -	227 -	TO -	Fail -	132	57	47

optimization and declare that a CLF has been found. To solve this bilinear problem, we use alternative minimization approach described below. First, V is initialized to be a positive definite function (by initializing $\mathbf{c}$ to some fixed value). Then, the approach repeatedly alternates between the following steps:

1. $\mathbf{c}$ is fixed, and we search for a θ that minimizes γ.
2. θ is fixed, and we search for a $\mathbf{c}$ that minimizes γ.

Each of these problems can be relaxed using Sum of Squares (SOS) programming [70]. The approach is iterated and results in a sequence of values $\gamma_0 \geq \gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_i$, wherein γ_i is the value of the objective after i optimization instances have been solved. Since the solution of one optimization instance forms a feasible solution for the subsequent instance, it follows that γ_i are monotonically nondecreasing. The iterations stop whenever γ does not decrease sufficiently between iterations. After termination, the approach succeeds in finding $V_{\mathbf{c}}$, K_θ only if $\gamma < 0$. Otherwise the approach fails.

Finding a suitable initial value for $\mathbf{c}$ is an important factor for success. As proposed by Majumdar et al, we pose and solve a linear feedback controller by applying the LQR method to the linearization of the dynamics [56]. In this case, we initialize V using the optimal cost function provided by the LQR. We also note that the linearization for the dynamics is not controllable for all cases and we can not always use this initialization trick. In these cases, we start from a initial candidate such as $||\mathbf{x}||_2^2$.

Additionally, Majumdar et al. (ibid) discuss solutions to handle input saturation for control inputs that must lie between two bounds. A precise approach considers 3^m different cases, where m is the number of control inputs to distinguish between each control input u_i being saturated at either limits or unsaturated. Furthermore, they provide a less expensive but conservative solution wherein they require $K_\theta(\mathbf{x}) \in U$ to avoid input saturation, which yields fewer constraints. Here, we consider three different variations of this method: (i) inputs are not saturated, (ii) inputs are saturated and the conservative solution is used, and (iii) inputs are saturated and the original/expensive solution is used. We consider variation (ii) only if the method is successful without forcing the input saturation, and we consider variation (iii) only if the conservative solution (variation (ii)) fails. For the Lyapunov function V we consider quadratic monomials as our basis functions, and for the feedback law K, we consider both linear and quadratic basis functions as separate problem instances. Similar to the SDP relaxation considered in this work, the SOS programming approach uses a degree limit D for the multiplier polynomials used in the positivstellensatz (cf. [48]). The limits used for the bilinear optimization approach are identical to those used in our method for each benchmark. The bilinear method is adapted to other properties used in our benchmarks and the results are shown in Table 4.

For the first two problem instances, the linearized dynamics are not controllable, and thus, we can not use the LQR trick for initialization. Instead, we use the solution obtained using our method as the starting point. Despite this, the bilinear optimization approach fails to find a feedback law. This suggests that the fixed structure of the feedback law K is more restrictive when compared to fixing a CLF and using Sontag's formula for synthesizing a feedback law K.

Table 4 Results for "bilinear formulation" method. K: basis functions used to parameterize K, L: basis functions are monomials with maximum degree 1 (linear), Q: basis functions are monomials with maximum degree 2 (quadratic), LQR: if LQR is used for initialization, ST.: saturation type, NP: numerical problem, St.: status.

Problem	Param.		Status		
System Name	K	LQR	ST.(i)	ST.(ii)	ST.(iii)
Unicycle-Seg. 2	L	✖	✖	-	-
	Q	✖	✖	-	-
Unicycle-Seg. 1	L	✖	✖	-	-
	Q	✖	✖	-	-
TORA	L	✓	✓	✓	-
Inverted Pend.	L	✓	✓	✓	-
Bicycle	L	✓	✓	✓	-
Bicycle*	L	✓	✓	✖	✓
Bicycle × 2	L	✓	NP	-	-
Forward Flight	L	✓	NP	-	-
Hover Flight	L	✓	✖	-	-
	Q	✓	✖	-	-

Bicycle*: The bicycle case-study where U is $[-5, 5]^2$ instead of $[-10, 10]^2$ (Our method could solve this problem instance as well).

For the remaining instances, we were able to use the LQR trick successfully to find an initial solution. Starting from this solution, the bilinear approach is successful on four problem instances, but fails for the hover flight problem. This suggests that even the LQR trick may not always provide a good initialization. For two of the larger problem instances, the bilinear method fails because of numerical errors, when dealing with large SDP problems. While the SOS programming has similar complexity compared to our method, it encounters numerical problems when solving large problems. We believe two factors are important here. First, our method solves different smaller verification problems and verifies each condition separately, while in a SOS formulation all conditions on V and ∇V are formulated into one big SDP problem. Moreover, in our method when we encounter a numerical error, we simply use the (potentially wrong) solution as a spurious counterexample without losing the soundness. Then, using demonstrations we continue the search. On the other hand, when the bilinear optimization procedure encounters a numerical error, it is unable to make further progress towards an optimal solution.

In conclusion, our method has several benefits when compared to the bilinear formulation. First, our method does not assume any specific parameterization for the feedback law. Instead it assumes a form for the CLF but uses Sontag's formula to obtain a feedback law. This is advantageous since we do not have to fix the structure of the feedback law in our approach. Second, our method uses demonstrations to generate a candidate instead of a local search, and we provide an upper-bound on the number of iterations. And finally, our method can sometimes recover from numerically ill-posed SDPs, and thus scales better as demonstrated through experiments. On the flip side, unlike the bilinear formulation, our method relies on a demonstrator that may not be easy to implement.

8 Related Work

In this section, we review the related work from the robotics, control, and formal verification communities.

Synthesis of Lyapunov Functions from Data: The problem of synthesizing Lyapunov functions for a control system by observing the states of the system in simulation has been investigated in the past by Topcu et al. to learn Lyapunov functions along with the resulting basin of attraction [97]. Whereas the original problem is bilinear, the use of simulation data makes it easier to postulate states that belong to the region of attraction, and therefore find Lyapunov functions that belong to this region by solving LMIs in each case. The application of this idea to larger black-box systems is demonstrated by Kapinski et al. [41], where the counterexamples are used to generate data iteratively. Our approach focuses on controller synthesis through learning a control Lyapunov function to replace an existing controller. A key difference lies in the fact that *we do not attempt to prove that the original demonstrator is necessarily correct*, but find a control Lyapunov function by assuming that the demonstrator is able to stabilize the system for the specific states that we query on. Another important contribution lies in our analysis of the convergence of the learning with a bound on the maximum number of queries needed. In fact, these results can also be applied to the Lyapunov function synthesis approaches mentioned earlier. Similar to our work, Khansari-Zadeh et al. [83] uses human demonstrations to generate data and enforce CLF conditions for the data points, to learn a CLF candidate. Their work does not include a verifier and therefore, the CLF candidate may not, in fact, be a CLF. However, the method can handle errors in the demonstrations by finding a maximal set of observations for which a compatible CLF exists, whereas our method does not address erroneous demonstrations.

Counter-Example Guided Inductive Synthesis: Our approach of alternating between a learning module that proposes a candidate and a verification module that checks the proposed candidate is identical to the counter-example guided inductive synthesis (CEGIS) framework originally proposed in verification community by Solar-Lezama et al. [86, 85]. As such, the CEGIS approach does not include a demonstrator that can be queried. The extension of this approach Oracle-guided inductive synthesis [39], generalizes CEGIS using an input/output *oracle* that serves a similar role as a demonstrator in this paper. However, the goal here is not to mimic the demonstrator, but to satisfy the specifications. Also, Jha et al. [40] prove bounds on the number of queries for discrete concept classes using results on exact concept learning in discrete spaces [32]. In this article, we consider searching over continuous concept class, and prove bounds on the number of queries under a robustness assumption.

The CEGIS procedure has been used for the synthesis of CLFs recently by authors [77, 79], combining it with SDP solvers for verifying CLFs. The key difference here lies in the use of the demonstrator module that simplifies the learning module. In the absence of a demonstrator module, the problem of finding a candidate reduces to solving linear constraints with disjunctions, an NP-hard problem [77]. Likewise, the convergence results are quite weak [78]. In the setting of this paper, however, the use of a MPC scheme as a demonstrator allows us to use faster LP solvers and provide convergence guarantees. Empirically, we are able to demonstrate the successful inference of CLFs on systems with up to eight state variables, whereas previous work in this space has been restricted to much smaller problems [77].

Learning from Demonstration: The idea of learning from demonstrations has a long history [5]. The overall framework uses a demonstrator that can, in fact, be a human operator [83, 43] or a complex MPC-based control law [90, 7, 81, 106, 60, 105]. The approaches differ on the nature of the interactions between the learner and the demonstrator; as well as how the policy is inferred. Our approach stands out in many ways: (a) We represent our policies by CLFs which are polynomial. On one hand, these are much less powerful than approaches that use neural networks [105], for instance. However, the advantage lies in our ability to solve verification problems to ensure that the resulting policy learned through the CLF is correct with respect to the underlying dynamical model. (b) Our framework is *adversarial.* The choice of the counterexample to query the demonstrator comes from a failed attempt to validate the current candidate. (c) Finally, we use simple yet powerful ideas from convex optimization to place bounds on the number of queries, paralleling some results on concept learning in discrete spaces [32].

Lyapunov Analysis for Controller Synthesis Sontag originally introduced Control Lyapunov functions and provided a universal construction of a feedback law for a given CLF [88, 89]. As such, the problem of learning CLFs is well known to be hard, involving bilinear matrix inequalities (BMIs) [94]. An more conservative (less precise) approach involves solving bilinear problems simultaneously for a control law and a Lyapunov function certifying it [25, 56]. This also leads to bilinear formulation. Prieur et al. [72] shows that the set of feasible solutions to such problem may not only be non-convex, but also disconnected. Nevertheless, there are some attempts to solve these BMIs which are well known to be NP-hard [35]. A common approach to solve these BMIs is to perform an alternating minimization by fixing one set of bilinear variables while minimizing over the other. Such an approach has poor guarantees in practice, often “getting stuck” on a saddle point that does not allow the technique to make progress in finding a feasible solution [33]. To combat this, Majumdar et al. (ibid) use LQR controllers and their associated Lyapunov functions for the linearization of the dynamics as good initial seed solutions [56]. In contrast, our approach simply assumes a demonstrator in the form of a MPC controller that can be used to resolve the bilinearity. Furthermore, our approach does not encounter the local saddle point problem. And finally, when the inputs are saturated, the complexity of such a method is exponential in the number of control inputs, while the complexity of our method remains polynomial.

Formal Controller Synthesis The use of the learning framework with a demonstrator distinguishes the approach in this paper from recently developed ideas based on formal synthesis. Majority of these techniques focus on a given dynamical system and a specification of the correctness in temporal logic to solve the problem of controller design to ensure that the resulting trajectories of the closed loop satisfy the temporal specifications. Most of these approaches are based on discretization of the state-space into cells to compute a discrete abstraction of the overall system [103, 54, 82, 62, 44]. Another set of solutions are based on formal parameter synthesis that search for unknown parameters so that the specifications are met [104, 23]. These methods include synthesize certificates (Lyapunov-like functions) by solving nonlinear constraints either through branch-and-bound techniques [36, 78], or through a combination of simulations and quantifier elimination [92, 93]. Our method is potentially more scalable, since the use of a demonstrator allows us to solve convex constraints instead. Raman et al. design a model-predictive control (MPC) from temporal logic properties [74]. More specifically, MILP solvers are used inside the MPC, which can be quite expensive for real-time control applications. We instead learn a CLF from the MPC and the CLF yields an easily computable feedback law (using Sontag’s formula).

Occupation Measures In this paper, we use the Lyapunov function approach to synthesizing controllers. An alternative is to use occupation measures [75,71,49,58]. These methods formulate an infinite dimensional problem to maximize the region of attraction and obtain a corresponding control law. This is relaxed to a sequence of finite dimensional SDPs [47]. Note however that the approach computes an over approximation of the finite time backward reachable set from the target and a corresponding control. Our framework here instead seeks an under-approximation that yields a guaranteed controller.

Modeling Inaccuracies and Safe Iterative Learning. A key drawback of our approach is its dependence on a mathematical model of the system for learning CLFs. Although this model is by no means identical to the real system, it is hoped that the CLF and the control law remain valid despite the unmodeled dynamics. Our recent work has successfully investigated physical experiments that use control Lyapunov-like functions learned from mathematical models for path following problems on a $\frac{1}{8}$-scale model vehicle using accurate indoor localization to obtain full state information in real-time [76]. The broader area of iterative learning controls considers the process of learning how to control a given plant at the same time as inferring a more refined model of the plant through exploration [29]. However, in order to avoid damaging the system, it is necessary to maintain the system state in a safe set while learning the system dynamics. Recent work by Wang et al. consider a combination of barrier certificates for maintaining safety while learning Gaussian process models of the vehicle dynamics [101]. Another approach considers safe reinforcement learning that incrementally refines a Gaussian process approximation of the unmodeled system dynamics, starting from a known initial model [11]. This approach uses a Lyapunov function and performs explorations at so-called "safe points" from which safety can be guaranteed during the exploration process. In doing so, the model of the system is updated along with an estimate of the safe set obtained as a region of attraction of the Lyapunov function.

9 Discussion and Future Work

In this section, we discuss some current limitations of our approach as well as possible extensions of our approach that can provide avenues for future research.

Extension to Switched Systems: Thus far, our focus has been on control affine systems. We note that a variation of our framework is applicable to switched systems. Specifically, one can transform a plant wherein the control is performed through switching between different modes into a problem over control affine systems. Let Q be a finite set of modes, such that the dynamics vary according the mode $q \in Q$ ($\dot{\mathbf{x}} = f_q(\mathbf{x})$). The controller is assumed to operate by selecting the current mode q of the plant. Then the condition on ∇V for stabilizing switched systems:

$$(\forall \mathbf{x} \neq \mathbf{0})\ (\exists q \in Q)\ \nabla V \cdot f_q(\mathbf{x}) < 0\,,$$

is replaced with

$$(\forall \mathbf{x} \neq \mathbf{0})\ (\exists \lambda \geq \mathbf{0}, \sum_q \lambda_q = 1)\ \sum_q \lambda_q \left(\nabla V \cdot f_q(\mathbf{x})\right) < 0.$$

This is identical to the conditions obtained for a control affine system, and thus, our framework can readily extend to such systems. Moreover, using the original formulation, checking conditions on ∇V is even simpler (compared to Eq. (17)):

$$(\exists \mathbf{x} \neq \mathbf{0})\ \bigwedge_q \nabla V \cdot f_q(\mathbf{x}) \geq 0\,.$$

Extensions to Discrete-Time Systems: Control problems on discrete-time systems have been widely studied. MPC schemes are naturally implemented over such systems, and furthermore, Lyapunov-like conditions extend quite naturally. As such, our approach can be extended to discrete-time nonlinear systems defined by maps as opposed to ODEs. However, polynomial discrete systems are known to pose computational challenges: when the Lie derivative is replaced by a difference operator, the degree of the resulting polynomial can be larger.

Optimizing Performance Criteria: Our approach stops as soon as one CLF is discovered. However, no claims are made as to the optimality of the CLF. The experimental results suggest that the controllers found by the CLFs are quite different from the original demonstrator in terms of their performance. An important extension to our work lies in finding CLFs so that the resulting controllers optimize some performance metric. One challenge lies in specifying these performance metrics as functions of the coefficients of the CLF. A simple approach may consist of using a black-box performance evaluation function over the CLF discovered by our approach. Once a CLF is found, we may continue our search but now target CLFs whose performance are strictly better than the ones discovered thus far.

Other Verifiers: The verifier is the main bottleneck in our learning framework. While in theory, the SDP relaxation addresses verification problems for polynomial system, the scalability for systems of high dimensions is still an issue. There are alternative solutions to the SDP relaxation, which promise better scalability. In particular linear relaxations are more attractive for this framework [2,10]. Using linear relaxations, one could restrict the candidate space to positive definite polynomials up front, and consider only the conditions over ∇V during the verification process. Therefore, using linear relaxations, not only the verification problem scales better, the number of such verifications to be solved can be decreased.

For a highly nonlinear system, the degree of polynomials for the dynamics as well as basis functions get larger. For these systems, the scalability is even more challenging. In future we wish to explore the the use of falsifiers (instead of verifiers) and move towards more scalable solutions [1,4,24]. While falsifiers would not guarantee correctness, they can be used to find concrete counterexamples. And by dropping formal correctness, a falsifier can replace the verifier in the learning framework.

Beyond Polynomial CLFs: In this paper, we assumed that the CLF candidate V is a linear combination of some given basis functions. While we showed that this model is precise enough to address exponential stability over compact sets, there are systems for which a smooth V does not exist. Nevertheless, our framework can also handle nonlinear models such as Gaussian mixture or feed forward neural network models, especially if the verifier is replaced by a falsifier that can be implemented through simulations. However, there are some serious drawbacks, including more expensive candidate generation, and weaker convergence guarantees. In future work we wish to investigate these models.

Beyond MPC-based Demonstrations: As mentioned earlier, the demonstrator is treated as a blackbox. We have investigated to use MPC as they are easy to design, and can provide smooth feedbacks which in our experiments is the key to find a smooth CLF. However, nonlinear MPC schemes using numerical optimization can guarantee convergence only to local minima, but this does not translate as such into guarantees of stability or that the original specifications are met. However, if we employed human demonstrators (for example, an expert who operates the system), the demonstrator may include errors, and we may need to consider approaches that can reject a subset of the given demonstrations [83]. In addition, the demonstrations can lead to inconsistent data, wherein nearby queries are handled using different strategies by the demonstrator, leading to no single CLF that is compatible with the given demonstrations [21,16]. These problems are left for future work.

10 Conclusion

We have thus proposed an algorithmic learning framework for synthesizing control Lyapunov-like functions for a variety of properties including stability, reach-while-stay. The framework provides theoretical guarantees of soundness, i.e., the synthesized controller is guaranteed to be correct by construction against the given plant model. Furthermore, our approach uses ideas from convex analysis to provide termination guarantees and bounds on the number of iterations.

Acknowledgements We are grateful to Mr. Sina Aghli, Mr. Souradeep Dutta, Prof. Christoffer Heckman and Prof. Eduardo Sontag for helpful discussions. This work was funded in part by NSF under award numbers SHF 1527075 and CPS 1646556. All opinions expressed are those of the authors and not necessarily of the NSF.

References

1. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivancic, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. Trans. on Embedded Computing Systems (TECS) **12**, 95– (2013)
2. Ahmadi, A.A., Majumdar, A.: Dsos and sdsos optimization: Lp and socp-based alternatives to sum of squares optimization. In: Information Sciences and Systems (CISS), 2014 48th Annual Conference on, pp. 1–5. IEEE (2014)
3. Ames, A.D., Powell, M.: Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs. In: Control of Cyber-Physical Systems, pp. 219–240. Springer (2013)
4. Annapureddy, Y.S.R., Liu, C., Fainekos, G.E., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: Tools and algorithms for the construction and analysis of systems, *LNCS*, vol. 6605, pp. 254–257. Springer (2011)
5. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems **57**(5), 469 – 483 (2009). DOI 10.1016/j.robot.2008.10.024
6. Artstein, Z.: Stabilization with relaxed controls. Nonlinear Analysis: Theory, Methods & Applications **7**(11), 1163 – 1173 (1983). DOI 10.1016/0362-546X(83)90049-4
7. Atkeson, C.G., Liu, C.: Trajectory-based dynamic programming. In: Modeling, Simulation and Optimization of Bipedal Walking, pp. 1–15. Springer (2013)
8. Atkinson, D.S., Vaidya, P.M.: A cutting plane algorithm for convex programming that uses analytic centers. Mathematical Programming **69**(1-3), 1–43 (1995). DOI 10.1007/BF01585551
9. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Springer (2003)
10. Ben Sassi, M.A., Sankaranarayanan, S., Chen, X., brahm, E.: Linear relaxations of polynomial positivity for polynomial lyapunov function synthesis. IMA Journal of Mathematical Control and Information **33**(3), 723–756 (2016). DOI 10.1093/imamci/dnv003

11. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 908–918. Curran Associates, Inc. (2017)
12. Bertsekas, D.P.: Dynamic programming and optimal control, vol. 1. Athena Scientific Belmont, MA (1995)
13. Bertsekas, D.P.: Approximate dynamic programming (2008)
14. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: A survey. Operations research **29**(6), 1039–1091 (1981). DOI 10.1287/opre.29.6.1039
15. Bouyer, P., Markey, N., Perrin, N., Schlehuber-Caissier, P.: Timed-automata abstraction of switched dynamical systems using control invariants. Real-Time Systems **53**(3), 327–353 (2017). DOI 10.1007/s11241-016-9262-3
16. Breazeal, C., Berlin, M., Brooks, A., Gray, J., Thomaz, A.L.: Using perspective taking to learn from ambiguous demonstrations. Robotics and Autonomous Systems **54**(5), 385 – 393 (2006). DOI https://doi.org/10.1016/j.robot.2006.02.004. The Social Mechanisms of Robot Programming from Demonstration
17. Brown, C.W., Davenport, J.H.: The complexity of quantifier elimination and cylindrical algebraic decomposition. In: Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, ISSAC '07, pp. 54–60. ACM, New York, NY, USA (2007). DOI 10.1145/1277548.1277557
18. Bryson, A.E.: Applied optimal control: optimization, estimation and control. CRC Press (1975)
19. Burridge, R.R., Rizzi, A.A., Koditschek, D.E.: Sequential composition of dynamically dexterous robot behaviors. The International Journal of Robotics Research **18**(6), 534–555 (1999)
20. Chan, R.P.M., Stol, K.A., Halkyard, C.R.: Review of modelling and control of two-wheeled robots. Annual Reviews in Control **37**(1), 89 – 103 (2013). DOI 10.1016/j.arcontrol.2013.03.004
21. Chernova, S., Veloso, M.: Learning equivalent action choices from demonstration. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pp. 1216–1221. IEEE (2008)
22. Curtis, J.W.: Clf-based nonlinear control with polytopic input constraints. In: 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475), vol. 3, pp. 2228–2233 Vol.3 (2003). DOI 10.1109/CDC.2003.1272949
23. Donzé, A., Krogh, B., Rajhans, A.: Parameter synthesis for hybrid systems with an application to simulink models. In: International Workshop on Hybrid Systems: Computation and Control, pp. 165–179. Springer (2009)
24. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: FORMATS, *Lecture Notes in Computer Science*, vol. 6246, pp. 92–106. Springer (2010)
25. El Ghaoui, L., Balakrishnan, V.: Synthesis of fixed-structure controllers via numerical optimization. In: Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on, vol. 3, pp. 2678–2683. IEEE (1994)
26. Elzinga, J., Moore, T.G.: A central cutting plane algorithm for the convex programming problem. Mathematical Programming **8**(1), 134–145 (1975). DOI https://doi.org/10.1007/BF01580439
27. Francis, B.A., Maggiore, M.: Models of mobile robots in the plane. In: Flocking and Rendezvous in Distributed Robotics, pp. 7–23. Springer (2016). DOI 10.1007/978-3-319-24729-8_2
28. Freeman, R., Kokotovic, P.V.: Robust nonlinear control design: state-space and Lyapunov techniques. Springer Science & Business Media (2008)
29. French, M., Rogers, E.: Non-linear iterative learning by an adaptive lyapunov technique. International Journal of Control **73**(10), 840–850 (2000)
30. Galloway, K., Sreenath, K., Ames, A.D., Grizzle, J.W.: Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs. IEEE Access **3**, 323–332 (2015)
31. Gao, S., Kong, S., Clarke, E.M.: dreal: An smt solver for nonlinear theories over the reals. In: International Conference on Automated Deduction, pp. 208–214. Springer (2013). DOI 10.1007/978-3-642-38574-2_14
32. Goldman, S., Kearns, M.: On the complexity of teaching. Journal of Computer and System Sciences **50**(1), 20 – 31 (1995). DOI 10.1006/jcss.1995.1003
33. Helton, J.W., Merino, O.: Coordinate optimization for bi-convex matrix inequalities. In: Proc. IEEE CDC, vol. 4, pp. 3609–3613 vol.4 (1997)
34. Henrion, D., Lasserre, J.B., Löfberg, J.: Gloptipoly 3: moments, optimization and semidefinite programming. Optimization Methods & Software **24**(4-5), 761–779 (2009)
35. Henrion, D., Lofberg, J., Kocvara, M., Stingl, M.: Solving polynomial static output feedback problems with penbmi. In: Proceedings of the 44th IEEE Conference on Decision and Control, pp. 7581–7586. IEEE (2005)
36. Huang, Z., Wang, Y., Mitra, S., Dullerud, G.E., Chaudhuri, S.: Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 7434–7439. IEEE (2015)
37. Jadbabaie, A., Hauser, J.: Control of a thrust-vectored flying wing: a receding horizon-lpv approach. International Journal of Robust and Nonlinear Control **12**(9), 869–896 (2002)
38. Jankovic, M., Fontaine, D., KokotoviC, P.V.: Tora example: cascade-and passivity-based control designs. IEEE Transactions on Control Systems Technology **4**(3), 292–297 (1996)
39. Jha, S., Gulwani, S., Seshia, S.A., Tiwari, A.: Oracle-guided component-based program synthesis. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10, pp. 215–224. ACM, New York, NY, USA (2010). DOI 10.1145/1806799.1806833
40. Jha, S., Seshia, S.A.: A theory of formal synthesis via inductive learning. Acta Informatica **54**(7), 693–726 (2017). DOI 10.1007/s00236-017-0294-5
41. Kapinski, J., Deshmukh, J.V., Sankaranarayanan, S., Arechiga, N.: Simulation-guided lyapunov analysis for hybrid dynamical systems. In: Proceedings of the 17th international conference on Hybrid systems: computation and control, pp. 133–142. ACM (2014)
42. Khachiyan, L.: An inequality for the volume of inscribed ellipsoids. Discrete & Computational Geometry **5**(1), 219–222 (1990). DOI 10.1007/BF02187786
43. Khansari-Zadeh, Mohammad, S., Khatib, O.: Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. Autonomous Robots **41**(1), 45–69 (2017). DOI 10.1007/s10514-015-9528-y

44. Kloetzer, M., Belta, C.: A fully automated framework for control of linear systems from temporal logic specifications. Automatic Control, IEEE Transactions on **53**(1), 287–297 (2008)
45. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings, pp. 282–293 (2006). DOI 10.1007/11871842_29
46. Landry, M., Campbell, S.A., Morris, K., Aguilar, C.O.: Dynamics of an inverted pendulum with delayed feedback control. SIAM Journal on Applied Dynamical Systems **4**(2), 333–351 (2005). DOI 10.1137/030600461
47. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM Journal on Optimization **11**(3), 796–817 (2001)
48. Lasserre, J.B.: Moments, positive polynomials and their applications. World Scientific (2009)
49. Lasserre, J.B., Henrion, D., Prieur, C., Trélat, E.: Nonlinear optimal control via occupation measures and lmi-relaxations. SIAM Journal on Control and Optimization **47**(4), 1643–1666 (2008)
50. Lavalle, S.M., Kuffner Jr, J.J.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions. Citeseer (2000)
51. Leth, T., Wisniewski, R., Sloth, C.: On the existence of polynomial lyapunov functions for rationally stable vector fields. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 4884–4889 (2017). DOI 10.1109/CDC.2017.8264381
52. Liberzon, D.: Switching in systems and control. Springer Science & Business Media (2012)
53. Lin, Y., Sontag, E.D.: A universal formula for stabilization with bounded controls. Systems & Control Letters **16**(6), 393 – 397 (1991). DOI 10.1016/0167-6911(91)90111-Q
54. Liu, J., Ozay, N., Topcu, U., Murray, R.M.: Synthesis of reactive switching protocols from temporal logic specifications. Automatic Control, IEEE Transactions on **58**(7), 1771–1785 (2013)
55. Lopez, I., McInnes, C.R.: Autonomous rendezvous using artificial potential function guidance. Journal of Guidance, Control, and Dynamics **18**(2), 237–241 (1995)
56. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control design along trajectories with sums of squares programming. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, pp. 4054–4061. IEEE (2013)
57. Majumdar, A., Tedrake, R.: Robust online motion planning with regions of finite time invariance. In: Algorithmic Foundations of Robotics X, pp. 543–558. Springer (2013). DOI 10.1007/978-3-642-36279-8_33
58. Majumdar, A., Vasudevan, R., Tobenkin, M.M., Tedrake, R.: Convex optimization of nonlinear feedback controllers via occupation measures. The International Journal of Robotics Research p. 0278364914528059 (2014)
59. Mason, M.: The mechanics of manipulation. In: Robotics and Automation. Proceedings. 1985 IEEE International Conference on, vol. 2, pp. 544–548. IEEE (1985)
60. Mordatch, I., Todorov, E.: Combining the benefits of function approximation and trajectory optimization. In: Proceedings of Robotics: Science and Systems. Berkeley, USA (2014). DOI 10.15607/RSS.2014.X.052
61. Mosek, A.: The mosek optimization software. Online at http://www. mosek. com **54**, 2–1 (2010)
62. Mouelhi, S., Girard, A., Gössler, G.: Cosyma: a tool for controller synthesis using multi-scale abstractions. In: Proceedings of the 16th international conference on Hybrid systems: computation and control, pp. 83–88. ACM (2013)
63. Nguyen, Q., Sreenath, K.: Optimal robust control for bipedal robots through control lyapunov function based quadratic programs. In: Robotics: Science and Systems (2015)
64. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer–Verlag (2006)
65. Papachristodoulou, A., Prajna, S.: On the construction of Lyapunov functions using the sum of squares decomposition. In: IEEE CDC, pp. 3482–3487. IEEE Press (2002)
66. Parillo, P.A.: Semidefinite programming relaxation for semialgebraic problems. Mathematical Programming Ser. B **96**(2), 293–320 (2003)
67. Peet, M.M.: Exponentially stable nonlinear systems have polynomial lyapunov functions on bounded regions. IEEE Transactions on Automatic Control **54**(5), 979–987 (2009)
68. Peet, M.M., Bliman, P.A.: Polynomial lyapunov functions for exponential stability of nonlinear systems on bounded regions. IFAC Proceedings Volumes **41**(2), 1111 – 1116 (2008). DOI 10.3182/20080706-5-KR-1001.00192. 17th IFAC World Congress
69. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: HSCC, vol. 2993, pp. 477–492. Springer (2004)
70. Prajna, S., Papachristodoulou, A., Parrilo, P.A.: Introducing sostools: A general purpose sum of squares programming solver. In: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, vol. 1, pp. 741–746. IEEE (2002)
71. Prajna, S., Parrilo, P.A., Rantzer, A.: Nonlinear control synthesis by convex optimization. IEEE Transactions on Automatic Control **49**(2), 310–314 (2004)
72. Prieur, C., Praly, L.: Uniting local and global controllers. In: Decision and Control, 1999. Proceedings of the 38th IEEE Conference on, vol. 2, pp. 1214–1219. IEEE (1999)
73. Primbs, J.A., Nevistić, V., Doyle, J.C.: Nonlinear optimal control: A control lyapunov function and receding horizon perspective. Asian Journal of Control **1**(1), 14–24 (1999)
74. Raman, V., Donzé, A., Sadigh, D., Murray, R.M., Seshia, S.A.: Reactive synthesis from signal temporal logic specifications. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, pp. 239–248. ACM (2015)
75. Rantzer, A.: A dual to lyapunov's stability theorem. Systems & Control Letters **42**(3), 161–168 (2001)
76. Ravanbakhsh, H., Aghli, S., Heckman, C., Sankaranarayanan, S.: Path-following through control funnel functions. CoRR **abs/1804.05288** (2018). URL http://arxiv.org/abs/1804.05288
77. Ravanbakhsh, H., Sankaranarayanan, S.: Counter-example guided synthesis of control lyapunov functions for switched systems. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 4232–4239 (2015). DOI 10.1109/CDC.2015.7402879
78. Ravanbakhsh, H., Sankaranarayanan, S.: Counterexample guided synthesis of switched controllers for reach-while-stay properties. arXiv preprint arXiv:1505.01180 (2015)

79. Ravanbakhsh, H., Sankaranarayanan, S.: Robust controller synthesis of switched systems using counterexample guided framework. In: 2016 International Conference on Embedded Software (EMSOFT), pp. 1–10 (2016). DOI 10.1145/2968478.2968485
80. Ravanbakhsh, H., Sankaranarayanan, S.: Learning lyapunov (potential) functions from counterexamples and demonstrations. In: Proceedings of Robotics: Science and Systems. Cambridge, Massachusetts (2017). DOI 10.15607/RSS.2017.XIII.049
81. Ross, S., Gordon, G.J., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: AISTATS, vol. 1, p. 6 (2011)
82. Rungger, M., Zamani, M.: Scots: A tool for the synthesis of symbolic controllers. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, pp. 99–104. ACM (2016)
83. S. Mohammad Khansari-Zadeh, Aude Billard: Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. Robotics and Autonomous Systems **62**(6), 752 – 765 (2014). DOI 10.1016/j.robot.2014.03.001
84. Shor, N.: Class of global minimum bounds on polynomial functions. Cybernetics **23**(6), 731–734 (1987). Originally in Russian: Kibernetika (6), 1987, 9–11
85. Solar-Lezama, A.: Program synthesis by sketching. ProQuest (2008)
86. Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., Saraswat, V.: Combinatorial sketching for finite programs. ACM SIGOPS Operating Systems Review **40**(5), 404–415 (2006)
87. Sontag, E.D.: A characterization of asymptotic controllability. In: Dynamical Systems II (Proc. of University of Florida International Symposium), pp. 645–648. Academic Press, NY (1982)
88. Sontag, E.D.: A lyapunov-like characterization of asymptotic controllability. SIAM Journal on Control and Optimization **21**(3), 462–471 (1983)
89. Sontag, E.D.: A 'universal' construction of artstein's theorem on nonlinear stabilization. Systems & Control Letters **13**(2), 117 – 123 (1989). DOI 10.1016/0167-6911(89)90028-5
90. Stolle, M., Atkeson, C.G.: Policies based on trajectory libraries. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pp. 3344–3349. IEEE (2006)
91. Suarez, R., Solis-Daun, J., Aguirre, B.: Global clf stabilization for systems with compact convex control value sets. In: Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), vol. 4, pp. 3838–3843 vol.4 (2001). DOI 10.1109/.2001.980463
92. Taly, A., Gulwani, S., Tiwari, A.: Synthesizing switching logic using constraint solving. International journal on software tools for technology transfer **13**(6), 519–535 (2011)
93. Taly, A., Tiwari, A.: Switching logic synthesis for reachability. In: Proceedings of the tenth ACM international conference on Embedded software, pp. 19–28. ACM (2010)
94. Tan, W., Packard, A.: Searching for control Lyapunov functions using sums of squares programming. In: Allerton conference on communication, control and computing, pp. 210–219 (2004)
95. Tarasov, S., Khachian, L., Erlikh, I.: The method of inscribed ellipsoids. Doklady Akademii Nauk. SSSR **298**(5), 1081–1085 (1988)
96. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: Lqr-trees: Feedback motion planning via sums-of-squares verification. The International Journal of Robotics Research (2010)
97. Topcu, U., Packard, A., Seiler, P., Wheeler, T.: Stability region analysis using simulations and sum-of-squares programming. In: Proceedings of the American control conference, pp. 6009–6014 (2007)
98. Vaidya, P.M.: A new algorithm for minimizing convex functions over convex sets. Mathematical programming **73**(3), 291–341 (1996). DOI 10.1007/BF02592216
99. Vandenberghe, L., Boyd, S., Wu, S.P.: Determinant maximization with linear matrix inequality constraints. SIAM journal on matrix analysis and applications **19**(2), 499–533 (1998). DOI 10.1137/S0895479896303430
100. Vanderbei, R.J.: Linear Programming: Foundations & Extensions (Second Edition). Springer (2001). Cf. http://www.princeton.edu/ rvdb/LPbook/
101. Wang, L., Theodorou, E.A., Egerstedt, M.: Safe learning of quadrotor dynamics using barrier certificates. CoRR **abs/1710.05472** (2017). URL http://arxiv.org/abs/1710.05472
102. Wieland, P., Allgower, F.: Constructive safety using control barrier functions. IFAC Proceedings Volumes **40**(12), 462 – 467 (2007). DOI 10.3182/20070822-3-ZA-2920.00076. 7th IFAC Symposium on Nonlinear Control Systems
103. Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., Murray, R.M.: Tulip: a software toolbox for receding horizon temporal logic planning. In: Proceedings of the 14th international conference on Hybrid systems: computation and control, pp. 313–314. ACM (2011)
104. Yordanov, B., Belta, C.: Parameter synthesis for piecewise affine systems from temporal logic specifications. In: International Workshop on Hybrid Systems: Computation and Control, pp. 542–555. Springer (2008)
105. Zhang, T., Kahn, G., Levine, S., Abbeel, P.: Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 528–535. IEEE (2016)
106. Zhong, M., Johnson, M., Tassa, Y., Erez, T., Todorov, E.: Value function approximation and model predictive control. In: 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), pp. 100–107. IEEE (2013)